“十二五”普通高等教育本科国家级规划教材

高等学校规划教材

Web 程序设计

（第 4 版）

吉根林　顾韵华　主编
吴军华　郑　玉　彭作民　编著

電子工業出版社
Publishing House of Electronics Industry
北京 · BEIJING

内 容 简 介

本书是“十二五”普通高等教育本科国家级规划教材，也是国家级精品课程和优秀教材建设成果。本书主要介绍 Web 程序设计的方法与技术，使读者学会建立网站。全书共 8 章，包括：Web 编程基础知识；Web 应用程序开发与运行环境 Dreamweaver MX 及 Visual Studio 2012；HTML 与 XML；层叠样式表 CSS；Web 客户端程序设计；Web 服务器端程序设计；Web 数据库程序设计；ASP.NET 综合应用实例。每章配有大量实例、习题和上机实验题及实验指导，并且免费提供 PPT 教学课件和程序源代码。

本书可作为高校计算机科学与技术、网络工程、软件工程、电子商务、信息管理与信息系统、现代教育技术等相关专业教材，也是 Web 程序开发人员实用的技术参考书。

图书在版编目（CIP）数据

Web 程序设计 / 吉根林，顾韵华主编，吴军华，郑玉，彭作民编著．—4 版．—北京：电子工业出版社，2015.7

ISBN 978-7-121-26036-0

Ⅰ. ①W… Ⅱ. ①吉… ②顾… ③吴… ④郑… ⑤彭… Ⅲ. ①网页制作工具—程序设计—高等学校—教材 Ⅳ. ①TP393.092

中国版本图书馆 CIP 数据核字（2015）第 097801 号

策划编辑：袁　玺　　文字编辑：戴晨辰
责任编辑：袁　玺
印　　刷：
装　　订：
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　邮编　100036
开　　本：787×1 092　1/16　印张：17.75　字数：454.4 千字
版　　次：2002 年 8 月第 1 版
　　2015 年 7 月第 4 版
印　　次：2015 年 7 月第 1 次印刷
定　　价：40.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

前　言

本书是“十二五”普通高等教育本科国家级规划教材，是国家级精品课程和优秀教材建设的成果。

《Web 程序设计》前 3 版，承蒙广大读者的支持，被几十所高校选为相关课程教材，至 2014 年 12 月已连续印刷 20 次。在教材出版后的几年中，Web 应用程序开发技术又有了新的发展，同时从服务教学、服务读者的角度看，该教材还需进一步完善。为此，有必要对本书再次进行修订。

本次修订根据我们近年来从事“Web 程序设计”教学的经验与体会及读者的反馈建议，坚持**“Web 程序设计”课程既定的教学目标：学会建网站；**在保持原书基本风格的基础上，根据 Web 应用程序开发技术的发展趋势，以 ASP.NET 程序设计为主线，介绍 Web 应用程序开发技术，并对部分章节的内容进行了调整优化，进一步提高本书的先进性和实用性。本次修订的具体情况如下：

（1）调整、优化本书的结构框架，以 ASP.NET 程序设计为主线介绍 Web 应用程序开发技术，从 Web 客户端程序设计、Web 服务器端程序设计、Web 数据库程序设计三个方面介绍 ASP.NET 程序设计的基本内容与相关技术。

（2）在第 3 版 ASP.NET 程序设计基本内容的基础上，全面介绍了 ASP.NET 程序开发所需的技术，并将 VB.NET 语言更换为 C#语言。

（3）删去了“ASP 程序设计”一章，不再介绍 ASP 的知识与技术。

（4）删去了 VBScript 脚本语言的介绍。

（5）在“Web 应用程序开发与运行环境”一章中，将 ASP.NET 应用程序开发工具 Visual Studio 2008 版升级为 Visual Studio 2012 版。

（6）在“ASP.NET 综合应用实例”一章中，删除了一个 ASP 综合应用实例，增加了一个 ASP.NET 综合应用实例，以培养读者对 ASP.NET 程序设计技术的综合应用能力。

本教材的**参考教学时数约为 90～100 学时**，其中**理论教学 60～64 学时**，**上机实验 36～40 学时**。全书**配有大量例题，还安排了上机实验题，并给出了实验指导**，包括实验目的、实验内容及实验步骤。其内容可能比教学时数所允许的稍多一些，可供教师讲课时选取或让学生自学。

本书为任课教师**提供 PPT 教学课件及例题源程序**，任课老师可在华信教育资源网 http://www.hxedu.com.cn **免费注册下载**。欢迎任课教师及时向我们反馈授课心得和建议。

本次修订，第 1、2 章分别由南京师范大学吉根林教授、彭作民副教授执笔；第 3、4、5 章由南京信息工程大学顾韵华教授执笔；第 6、7 章分别由南京工业大学郑玉副教授、吴军华副教授执笔；第 8 章由顾韵华和彭作民共同编写；全书由吉根林和顾韵华担任主编，并统稿、定稿。在本次修订过程中，得到了电子工业出版社的支持，在此表示衷心的感谢！

由于作者水平有限，本书还会存在错误与不足之处，恳请广大读者与同行给予批评指正。作者 E-mail 地址：glji@njnu.edu.cn。

作　者

目　　录

第 1 章　Web 编程基础知识

本章介绍开发 Web 程序应具备的基础知识，包括 Web 的基本概念和工作原理、Internet 网络协议、IP 地址、域名和统一资源定位器 URL、ASP、ASP.NET、PHP、JSP 等动态网页设计技术以及.NET 框架，为在本课程中学习 Web 程序设计方法和开发技术做好准备。

1.1　什么是 Web

现在 Internet 已成为世界上最大的信息宝库，然而 Internet 上的信息资源既没有统一的目录，也没有统一的组织和系统，这些信息分布在 Internet 位于世界各地的计算机系统中。人们为了充分利用 Internet 上的信息资源，迫切需要一种方便快捷的信息浏览和查询工具，在这种情况下，Web 诞生了。

Web，全称为 World Wide Web，缩写为 WWW。Web 有许多译名，如环球网、万维网、全球信息网等。如果有一台计算机与 Internet 相连，不管它是通过什么方式连入 Internet 的，任何人都可以通过浏览器（Browser）访问处于 Internet 上任何位置的 Web 站点。但什么是 Web，目前尚无公认的准确定义。简单地说，Web 是一种体系结构，通过它可以访问分布于 Internet 主机上的链接文档。这一说法包含以下几层含义。

（1）Web 是 Internet 提供的一种服务。尽管这几年 Web 的迅猛发展使得有人甚至误认为 Web 就是 Internet，但事实上，Web 是基于 Internet、采用 Internet 协议的一种体系结构，因而它可以访问 Internet 的每一个角落。

（2）Web 是存储在全世界 Internet 计算机中的、数量巨大的文档的集合。或者可以通俗地说，Web 是世界上最大的电子信息仓库。

（3）Web 上的海量信息是由彼此关联的文档组成的，这些文档称为主页（Home Page）或页面（Page），它是一种超文本（Hypertext）信息，而使其连接在一起的是超链接（Hyperlink）。由于超文本的特性，用户可以看到文本、图形、图像、视频、音频等多媒体信息，这些媒体称为超媒体（Hypermedia）。

（4）Web 的内容保存在 Web 站点（Web 服务器）中，用户可通过浏览器访问 Web 站点。因此 Web 是一种基于浏览器/服务器（Browser/Server，简称 B/S）的结构。也就是说，Web 实际上是一种全球性通信系统，它通过 Internet 使计算机相互传送基于超媒体的数据信息。

（5）Web 以一些简单的操作方式（如单击鼠标）连接全球范围的超媒体信息。因此，它易于使用和普及。基于 Web 开发的各种应用易于跨平台实现，开发成本较低，而且基于 Web 的应用几乎不需要培训用户。

近年来，Web 得到了迅猛的发展，如今的 Web 应用已远远超出了原先对它的设想。它不仅成为 Internet 上最普遍的应用，而且正是由于它的出现，使 Internet 普及推广的速度大大提高了。

Web 具有以下特点。

（1）Web 是一种超文本信息系统。Web 的超文本链接使得 Web 文档不再像书本一样是固定的、线性的，而是可以从一个位置迅速跳转到另一个位置，从一个主题迅速跳转到另一个相关主题。

（2）Web 是图形化的和易于导航的。Web 之所以能够迅速流行，一个很重要的原因就在于它

具有在一页上同时显示图形、图像和其他超媒体的性能。在 Web 之前，Internet 上的信息只有文本形式，Web 则提供将图形、图像、音频、视频信息集于一体的特性。同时，Web 是非常易于导航的，只需要从一个链接跳转到另一个链接，就可以在各页面、各站点之间进行浏览了。

（3）Web 与平台无关。无论系统的软、硬件平台是什么，都可以通过 Internet 访问 WWW。Web 对系统平台没有限制。

（4）Web 是分布式的。对于 Web，没有必要把大量图形、图像、音频、视频信息都放在一起，可以将它们放在不同的站点上，只要通过超链接指向所需的站点，就可以使存放在不同物理位置上的信息实现逻辑上的一体化。对用户来说，这些信息是一体的。

（5）Web 具有新闻性。Web 站点上的信息是动态的、经常更新的。信息的提供者可以经常对站点上的信息进行更新，所以用户（浏览者）可以得到最新的信息。

（6）Web 是动态的、交互的。早期的 Web 页面是静态的，用户只能被动浏览。由于开发了多种 Web 动态技术，现在的用户已经能够方便地定制页面。以 ASP（Active Server Pages，动态服务器页面）、ASP.NET 和 Java 为代表的动态技术使 Web 从静态的页面变成可执行的程序，从而大大提高了 Web 的动态性和交互性。Web 的交互性还表现在它的超链接上，因为通过超链接，用户的浏览顺序和所到站点完全可由用户自行决定。

1.2 Web 的工作原理

Web 是一种典型的基于浏览器/服务器（Browser/Server，简称 B/S）的体系结构。典型的 B/S 结构将计算机应用分成三个层次，即客户端浏览器层、Web 服务器层和数据库服务器层。B/S 结构有许多优点，它简化了客户端的维护，所有的应用逻辑都是在 Web 服务器上配置的。B/S 结构突破了传统客户机/服务器（Client/Server，简称 C/S）结构中局域网对计算机应用的限制，用户可以在任何地方登录 Web 服务器，按照用户角色执行自己的业务流程。Web 通过 HTTP 协议实现客户端浏览器和 Web 服务器的信息交换，其基本工作原理如图 1-1 所示。

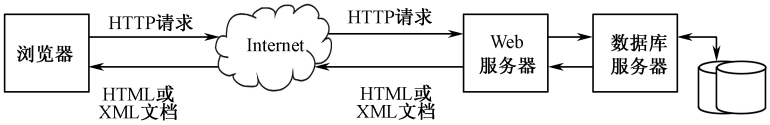


图 1-1　Web 的基本工作原理

Web 浏览器是一种 Web 客户端程序，用户要浏览 Web 页面，必须在本地计算机中安装浏览器软件。通过在浏览器地址栏中输入 URL 资源地址，将 Web 服务器中特定的网页文件下载到客户端计算机中，并在浏览器中打开。因此，从本质讲，浏览器是一种特定格式的文档阅读器，它能根据网页内容，对网页中的各种标记进行解释显示；同时，浏览器也是一种程序解释机，如果网页中包含客户端脚本程序，那么浏览器将执行这些客户端脚本代码，从而增强网页的交互性和动态效果。

在 Web 系统中，Web 服务器有两个层面的含义：一是指安装了 Web 服务程序的计算机；二是指 Web 服务器程序，可以管理各种 Web 文件，并为提出 HTTP（HyperText Transfer Protocol，超文本传输协议）请求的浏览器提供 HTTP 响应。要使一台计算机成为一台 Web 服务器，需要配置服务器操作系统，如 UNIX、Windows、Linux 等网络操作系统，并且还要安装专门的信息服务器程序，如 Windows 所提供的 IIS（Internet Information Server，互联网信息服务）。大多数情况下，Web 服务器和浏览器处于不同的机器中，但它们也可以并存在同一台机器中。

Web 服务器向浏览器提供服务的过程大致可以归纳为以下步骤。

（1）用户打开计算机（客户机），启动浏览器程序（Netscape Navigator、Microsoft Internet Explorer 等），并在浏览器中指定一个 URL（Uniform Resource Locator，统一资源定位器），浏览器便向该 URL 所指向的 Web 服务器发出请求。

（2）Web 服务器（也称 HTTP 服务器）接到浏览器的请求后，把 URL 转换成页面所在服务器的文件路径名。

（3）如果 URL 指向的是普通的 HTML（Hypertext Markup Language，超文本标记语言）文档，Web 服务器将直接把它传送给浏览器。HTML 文档中可能包含用 Java、JavaScript、ActiveX、VBScript、C#等编写的小应用程序（Applet），服务器也将它们随 HTML 文档一道传送给浏览器，在浏览器所在的机器中执行。

（4）如果 HTML 文档中嵌有 ASP 或 ASP.NET 程序，那么 Web 服务器就运行 ASP 或 ASP.NET 程序，并将结果传送给浏览器。Web 服务器运行 ASP 或 ASP.NET 程序时，还可能调用数据库服务器和其他服务器。

（5）URL 也可以指向 VRML（Virtual Reality Modeling Language，虚拟现实建模语言）文档。只要浏览器中配置有 VRML 插件，或者客户机中已安装 VRML 浏览器，就可以接收 Web 服务器发送的 VRML 文档。

早期的 Web 页面是静态的，用户只能被动浏览。静态页面是用纯 HTML 代码编写的，这些页面的代码保存为.html 或.htm 文件形式。后来，以 ASP、ASP.NET 和 Java 为代表的动态技术使 Web 从静态页面变成可执行的程序，从而产生了动态网页，大大提高了 Web 的动态性和交互性。利用 ASP 或 ASP.NET，服务器可以执行用户用 VBScript、JavaScript 或 C#编写的嵌入 HTML 文档中的程序。通过 ASP 或 ASP.NET 程序，Web 页面可以访问数据库，存取服务器的有关资源，使 Web 页面具有强大的交互能力。Web 的交互性还表现在它的超链接上，因为通过超链接，用户的浏览顺序和所到站点完全可由用户自行决定。

随着技术的不断发展，动态网页的实现一般采用客户端编程和服务器端编程两种程序设计方法。

（1）客户端编程就是客户端浏览器下载服务器中的程序来执行有关动态服务工作。程序员把客户端代码编写到 HTML 文件中，当用户提出对某个网页的请求时，这些客户端代码和 HTML 文件代码一起以响应方式返回提出请求的浏览器。常见的客户端编程技术有 VBScript、JavaScript、Java Applet 等。

（2）服务器端编程就是将程序员编写的代码保存在服务器中，当用户提出对某个网页的请求时，这个请求所要访问的页面代码都在服务器端执行，并把执行结果以 HTML 文件代码的形式传回浏览器，这样浏览器接收的只是程序执行的结果。常见的服务器端编程技术有 PHP、JSP、ASP、ASP.NET。

1.3　Internet 网络协议

Internet 是由各种不同类型、不同规模、独立管理和运行的主机或计算机网络组成的一个全球性特大网络。Internet 使用的网络协议是 TCP/IP 协议，凡是连入 Internet 的计算机都必须安装和运行 TCP/IP 协议软件。

1.3.1　TCP/IP 协议

TCP/IP 协议是一个协议集，其中最重要的是 TCP 协议和 IP 协议，因此，通常将这些协议简称为 TCP/IP 协议。

TCP/IP 协议把整个网络分成 4 个层次：应用层、传输层、网络层和物理链路层。它们都建立

在硬件基础之上。图 1-2 给出了 OSI 参考模型与 TCP/IP 参考模型的对照。

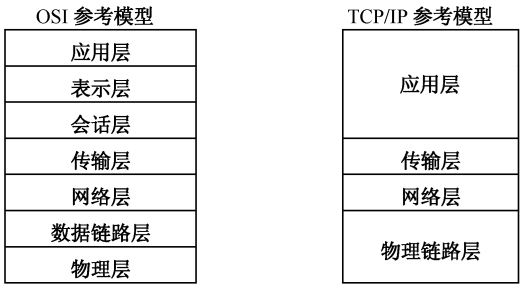


图 1-2　OSI 参考模型与 TCP/IP 参考模型的对照

（1）应用层。它是 TCP/IP 参考模型的最高层，向用户提供一些常用应用程序，如电子邮件服务等。应用层包括所有的高层协议，并且总是不断有新的协议加入。应用层主要有以下协议：

① 网络终端协议 Telnet　　//用于实现互联网中的远程登录功能；
② 文件传输协议 FTP　　//用于实现互联网中交互式文件传输功能；
③ 简单电子邮件协议 SMTP　　//用于实现互联网中电子邮件收发功能；
④ 网络文件系统 NFS　　//用于网络中不同主机间的文件系统共享；
⑤ 域名服务系统 DNS　　//用于实现网络设备域名到 IP 地址的映射服务；
⑥ 超文本传输协议 HTTP　　//用于在 Web 浏览器和服务器之间传输 Web 文档。

（2）传输层。传输层也叫 TCP 层，主要功能是负责应用进程之间的端-端通信。传输层定义了两种协议：传输控制协议 TCP 和用户数据报协议 UDP。

（3）网络层。网络层也叫 IP 层，负责处理互联网中计算机之间的通信，向传输层提供统一的数据报文。它的主要功能包括以下三个方面：

① 处理来自传输层的分组发送请求；
② 处理接收的数据报文；
③ 处理互连的路径。

（4）物理链路层。它的主要功能是接收 IP 层的 IP 数据报文，通过网络向外发送；接收并处理从网络上传来的物理帧，抽出 IP 数据报文，向 IP 发送。该层是主机与网络的实际连接层。

1.3.2　HTTP 协议

HTTP 是专门为 Web 设计的一种网络协议，它属于 TCP/IP 参考模型中的应用层协议，位于 TCP/IP 协议的顶层。因此，它在设计和使用中以 TCP/IP 协议集中的其他协议为基础。例如，它要通过 DNS 进行域名与 IP 地址的转换，要建立 TCP 链接才能进行文档传输。

Web 浏览器和服务器用 HTTP 协议来传输 Web 文档。HTTP 基于客户端请求、服务器响应的工作模式，其定义的事务处理由以下 4 个步骤组成：

（1）客户端与服务器建立连接；
（2）客户端向服务器提出请求；
（3）如果请求被接受，则服务器送回响应，在响应中包括状态码和所需的文件；
（4）客户端和服务器断开连接。

1.3.3　远程登录协议 Telnet

Telnet 是关于远程登录的一个协议。要使用 Telnet，在用户的计算机中需要安装和运行一个名

为 Telnet 的程序。在使用 Telnet 时，它又是一个命令。用户可以用 Telnet 命令使用户主机连入 Internet 上任何一台 Telnet 服务器。一般把这台被用户主机调用的服务器称为远程主机。这时候用户主机就成为该远程主机的一个终端。不管这种连接如何复杂，在用户的 PC 键盘上输入一个 Telnet 子命令后，总能在远程主机上得到服务响应，并把结果送回到用户的 PC 屏幕上。

Internet 上存在成千上万的各种主机（大、中、小型机）或服务器。用户可以通过 Telnet 连入某个主机并成为该主机的终端，进而用户可访问所需的各种信息，或运行远程主机中的程序来求解各种复杂的问题，一切都是在远程主机中快速执行（而不是将程序调回到用户主机中执行）后再从远程主机返回服务的结果。用户还可以利用 Telnet 连到 Internet 的各种服务器，如 Archie、Gopher、Wais、WWW 及其他服务器，比如某图书馆的资料文献服务器等。

用户使用远程主机有两种情况：一种是要求用户有账号才能登录的；另一种是开放的，用户无须拥有自己的账号，即不用口令和用户名就能登录。在 Internet 上有许多这样的为公众开放的 Telnet 远程服务。

1.3.4　文件传输协议 FTP

Telnet 让用户主机能以终端方式共享 Internet 上各类主机的资源，却不能把远程主机中的文件复制到用户主机中。有了 FTP 的帮助就能使 Internet 上两台主机间互传（复制）文件。FTP 有一套独立通用的命令（子命令），命令风格与 DOS 命令相似，如 Dir 为显示目录/文件。实际使用 FTP 时往往会碰到两个难点。第一，并不知道想要复制的文件在哪个 FTP 服务器中，在成千上万个 FTP 服务器中一个个地寻找某个文件犹如大海捞针。此时需要借助某些工具，如 Internet 上的 Archie 服务器。第二，要明确传送的文件是什么类型的，即确定传送的是二进制文件还是 ASCII 码文件。如果文件传送类型不对，复制得到的文件常常是无用的文件。

FTP 既是一种文件传输协议，也是一种服务。提供这种服务的设施叫作 FTP 服务器。有一种 FTP 服务器称为匿名 FTP 服务器，用户无须拥有口令和用户名就能与匿名 FTP 服务器实现连接并复制文件。在 Internet 上有许多这样的、为公众开放的匿名 FTP 服务器。

1.4　IP 地址、域名和 URL

1.4.1　IP 地址

IP 地址是识别 Internet 中主机及网络设备的唯一标识。每个 IP 地址通常分为网络地址和主机地址两部分，其长度为 4 B（字节），共 32 位，由 4 个用“ . ”分隔的十进制数组成，每个数不大于 255，如 202.119.106.253。

IP 地址可分成 5 类，其中常用的是如下 3 类。

A 类：用于规模很大、主机数目非常多的网络。A 类地址的最高位为 0，接下来的 7 位为网络地址，其余 24 位为主机地址。A 类地址允许组成 126 个网络，每个网络可包含 1 700 万台主机。

B 类：用于中型和大型网络。B 类地址最高两位为 10，接下来 14 位为网络地址，其余 16 位为主机地址。B 类地址允许组成 16 384 个网络，每个网络可包含 65 000 台主机。

C 类：用于小型本地网络（LAN）。C 类地址最高 3 位为 110，接下来 21 位为网络地址，其余 8 位为主机地址。

注意，主机地址的末字节不能取 0 和 255 两个数。

1.4.2 域名

IP 地址是连网计算机的地址标识，但对大多数人来说，记住很多计算机的 IP 地址并不是一件容易的事，所以 TCP/IP 协议中提供了域名服务系统（DNS），允许为主机分配字符名称，即域名。在网络通信时由 DNS 自动实现域名与 IP 地址的转换。例如，南京师范大学 Web 服务器的域名为 www.njnu.edu.cn。

Internet 中的域名采用分级命名，其基本结构如下：

```
计算机名.三级域名.二级域名.顶级域名
```

域名的结构与管理方式如下。

首先，DNS 将整个 Internet 划分成多个域，称为顶级域，并为每个顶级域规定了国际通用的域名。顶级域名采用两种划分模式，即组织模式和地理模式。有 7 个域对应于组织模式，其余的域对应于地理模式，如 cn 代表中国，us 代表美国，jp 代表日本等。

7 个组织模式的顶级域名分配如下：

```
com     //商业组织；
edu     //教育机构；
gov     //政府部门；
mil     //军事部门；
net     //网络中心；
org     //上述以外的组织；
int     //国际组织。
```

其次，Internet 的域名管理机构将顶级域的管理权分派给指定的管理机构，各管理机构对其管理的域继续进行划分，即划分成二级域，并将二级域的管理权授予其下属的管理机构，依此类推，便形成了树形域名结构。由于管理机构是逐级授权的，所以最终的域名都得到了 Internet 的承认，成为 Internet 中的正式名字。

1.4.3 统一资源定位器 URL

WWW 信息分布在全球，要找到所需信息就必须有一种说明该信息存放在哪台计算机的哪个路径下的定位信息。URL 就是用来确定某信息位置的方法。

URL 的概念实际上并不复杂，就像指定一个人要说明他的国别、地区、城镇、街道、门牌号一样，URL 指定 Internet 资源位于哪台计算机的哪个目录中。URL 通过定义资源位置的抽象标识来定位网络资源，其格式如下：

```
<信息服务类型>：//<信息资源地址>/<文件路径>
```

<信息服务类型>是指 Internet 的协议名，包括 ftp（文件传输服务）、http（超文本传输服务）、gopher（Gopher 服务）、mailto（电子邮件地址）、telnet（远程登录服务）、news（提供网络新闻服务）和 wais（提供检索数据库信息服务）。

<信息资源地址>指定一个网络主机的域名或 IP 地址。在有些情况下，主机域名后还要加上端口号，域名与端口号之间用冒号隔开。这里的端口是指操作系统用来辨认特定信息服务的软件端口。一般，服务器程序采用标准的保留端口号，因此用户在 URL 输入中可以省略它们。以下是一些 URL 的例子：

```
http: //www.njnu.edu.cn
http: //www.whitehouse.gov
telnet: //odysseus.circe.com:70
ftp: //ftp.w3.org/pub/www/doc
gopher: //gopher.internet.com
```

```
news: //comp.sys.novell
wais: //quake.think.com/directory-of-servers
```

1.5　动态网页设计技术简介

随着网络技术的不断发展，单纯的静态网页已经远远不能满足 Internet 发展的需要。早期，动态网页使用的主要是 CGI（Common Gateway Interface，公共网关接口）技术，可以使用不同的语言编写合适的 CGI 程序，如 Visual Basic、C/C++等。虽然 CGI 技术已经发展成熟且功能强大，但由于编程困难、效率较低、修改复杂等缺陷，因此 CGI 技术已被淘汰。

ASP 是一种功能强大的服务器端脚本编程环境，它是微软公司的产品，从 Windows NT Server 操作系统开始就附带这种脚本编程环境。1996 年底，微软公司推出了 ASP1.0，它内含于 IIS3.0 之中。1998 年，微软推出了 ASP2.0。2000 年，微软公司发布了 Windows 2000 操作系统，这个版本给我们带来了 IIS5.0 和 ASP3.0。ASP 最大的好处是可以包含 HTML 标记，也可以直接存取数据库以及使用 ActiveX 控件，它采用脚本语言 VBScript、JavaScript 作为开发语言，利用 HTML 网页、ASP 指令和 ActiveX 组件建立动态、交互的 Web 服务器应用程序。由于 ASP.NET 的出现，与 ASP 相比，ASP.NET 在功能、效率等方面都具有优势，因此，目前 ASP 基本不再使用。

目前比较受关注的动态网页设计技术主要有 PHP、JSP、ASP.NET。

1.5.1　PHP

PHP（Hypertext Preprocessor，超文本预处理器）是一种跨平台的服务器端嵌入式脚本语言，它是一种易于学习和使用的服务器端脚本语言，嵌入 HTML 文件，大量地借用 C、Java 和 Perl 语言的语法，并耦合 PHP 本身的特性，形成了自己的独特风格。PHP 支持目前绝大多数的数据库，Web 开发者使用 PHP 能够快速地写出生成动态网页的脚本代码。PHP 是完全免费的，可以从 PHP 官方网站（http://www.php.net）自由下载，可以不受限制地获得源代码，并可加入自己需要的功能。

PHP 具有如下特点。

（1）支持多种系统平台，包括 Windows、UNIX 和 Linux 系统。

（2）强大的数据库操作功能。PHP 提供丰富的数据库操作函数，它为各种流行数据库，包括 Linux 平台的 PostgreSQL、MySQL、Solid 及 Oracle，Windows 平台的 SQL Server，都设计了专门的函数，使操作这些数据库十分方便。

（3）易于与现有的网页融合。与 ASP、JSP 一样，PHP 也可结合 HTML 语言共同使用；它与 HTML 语言具有非常好的兼容性，使用者可以直接在脚本代码中加入 HTML 标记，或者在 HTML 标记中加入脚本代码从而更好地实现页面控制，提供更加丰富的功能。

（4）具有丰富的功能。PHP 提供结构化特性、面向对象设计、数据库处理、网络接口使用及安全编码机制等全面的功能。

（5）可移植性好。只需要进行很少的修改就可将整个网站从一个平台移植到另一个平台上，如从 Windows 平台移植到 UNIX 平台。

1.5.2　JSP

JSP（Java Server Pages，Java 服务器页面）是 Sun 公司于 1999 年 6 月推出的网站开发语言。它是基于 Java Servlet 及整个 Java 体系的 Web 开发技术，利用这一技术可以建立先进、安全和跨平台的动态网站。它完全解决了目前 ASP、PHP 的一个通病——脚本级执行。

JSP 与 ASP 在技术方面有许多相似之处。两者都是为实现 Web 动态交互网页制作而提供的技

术支持环境，都能帮助程序开发人员实现应用程序的编制与自带组件的网页设计，都能替代 CGI 使网站建设与发展变得简单又快捷。由于它们来源于不同的技术规范，因而其实现的基础不同，即对 Web 服务器平台的要求不同。基于 JSP 技术的应用程序比基于 ASP 的应用程序更易于维护和管理。

JSP 技术具有以下优点。

（1）内容生成与显示分离。使用 JSP 技术，Web 页面开发人员可以使用 HTML 或 XML 标记来设计页面。使用 JSP 标记或小脚本来生成页面上的动态内容（内容是动态的，但可根据用户请求而变化）。动态生成的内容被封装在标记和 JavaBeans 组件中，并且捆绑在小脚本中，所有的脚本在服务器端运行。

在服务器端，使用 JSP 引擎来解释 JSP 标记和小脚本，生成所请求的内容，并将结果以 HTML 或 XML 页面形式发送回浏览器。这有助于作者保护自己的代码，又能保证任何基于 HTML 的 Web 浏览器的完全可用性。

（2）可重用的组件。绝大多数 JSP 页面依赖于可重用的、跨平台的组件来执行应用程序所要求的复杂处理，如使用 JavaBeans 或 Enterprise JavaBeansTM 组件。开发人员可以共享各种组件，这种基于组件的方法提高了系统的开发效率。

（3）采用标记简化页面开发。JSP 技术使用 XML 标记封装了许多与动态内容生成相关的功能，页面开发人员使用这些标记就可以进行设计，而不必进行编程。

（4）适应更广泛的平台。JSP+JavaBean 可以在大多数 Web 服务器平台下使用。著名的 Web 服务器 Apache 能够很好地支持 JSP，由于 Apache 广泛应用在 Windows、UNIX 和 Linux 操作系统上，因此 JSP 有更广泛的运行平台。

（5）易于连接数据库。Java 中连接数据库的技术是 JDBC（Java DataBase Connectivity，Java 数据库连接）。很多数据库系统，如 Oracle、Sybase、MS SQL Server 和 MS Access 等，都带有 JDBC 驱动程序，Java 程序通过 JDBC 驱动程序与数据库相连，执行查询数据、提取数据等操作。另外，Sun 公司还开发了 JDBC-ODBC bridge，使用此项技术，Java 程序就可以访问带有 ODBC 驱动程序的数据库了。

1.5.3 ASP.NET

ASP.NET 是微软公司于 2001 年推出的一种用于创建 Web 应用程序的编程模型。它在结构上与前面的版本大不相同，它几乎完全是基于组件和模块化的。Web 应用程序的开发人员使用这个开发环境可以实现更加模块化、功能更强大的应用程序。

在 ASP.NET 中，所有程序保存在服务器端，由服务器编译执行。当第一次执行一个程序时进行编译，当再次执行这个程序时，就在服务器端直接执行它的已编译好的程序代码，因而 ASP.NET 程序的执行速度有较大的提高。对于实现同样功能的程序，ASP.NET 使用的代码量比 ASP 要小得多。ASP.NET 采用全新的编程环境，代表了技术发展的主流方向。从深层次说，ASP.NET 与 ASP 的主要区别体现在以下三个方面。

（1）效率。ASP 是一个脚本编程环境，只能用 VBScript 或 JavaScript 这样的非模块化语言来编写。当 ASP 程序完成之后，在每次请求时都要解释执行。这就意味着，它在使用其他语言编写大量组件的时候会遇到困难，并且无法实现对操作系统的底层操作。ASP.NET 则是建立在.NET 框架之上的，它可以使用 Visual Basic、C#、J#这样的模块化程序设计语言，并且它在第一次执行时进行编译，之后的执行不需要重新编译就可以直接运行，所以速度和效率比 ASP 提高很多。

（2）可重用性。在编写 ASP 应用程序时，ASP 代码和 HTML 混合在一起。只要需要，就可以在任意的位置插入一段代码来实现特定的功能。这种方法表面上看起来很方便，但实际上会产生

大量烦琐的页面，很难让人读懂，导致代码维护困难。ASP.NET 则可以实现代码和内容的完全分离，使得维护更方便。

（3）代码量。ASP 对所有要实现的功能均需要通过编写代码来实现。例如，为了保证一个用户数据提交页面的友好性，当用户输入错误时应显示错误的位置，并尽量把用户原来的输入显示在控件中。对于这样一个应用，使用 ASP 需要程序员编写大量的代码才能实现。在 ASP.NET 中，程序员只要预先说明，ASP.NET 就可以自动实现这样的功能。所以相对来说，要实现同样的功能，使用 ASP.NET 比使用 ASP 的代码量要小得多。

1.6　.NET 框架简介

.NET 是微软公司提出的新一代程序开发框架，而 ASP.NET 属于.NET 框架的一部分，是.NET 框架的一个应用模型，运行于具有.NET 框架环境的服务器中，可以使用多种语言开发，主要用于创建 Web 应用程序、网站及 Web 服务。

.NET 框架（.NET Framework）主要分如下为 4 个部分。

1．通用语言开发环境

开发程序时，如果使用符合通用语言规范的开发语言，那么开发的程序可以在任何有通用语言开发环境的操作系统下运行，包括 Windows NT/2000/XP 等。

2．.NET 基础类库

.NET 基础类库是一套函数库，以结构严密的树形结构组织，并由命名空间和类组成，功能强大，使用简单，具有高度的可扩展性。

3．.NET 开发语言

.NET 是多语言开发平台，微软公司最初提供了 5 种语言：VB.NET、JScript.NET、J#.NET、Managed C++.NET 及 C#。其他厂商还提供了很多对.NET 的语言支持，包括 COBOL、Eiffel、Perl、Pythn、Smalltalk、Scheme 等。

4．Visual Studio.NET 集成开发环境

Visual Studio.NET 集成开发环境是开发.NET 应用的利器，功能非常强大。

本 章 小 结

本章主要介绍了 Web 编程的基础知识，包括 Web 的基本概念和工作原理、Internet 网络协议、IP 地址、域名和统一资源定位器 URL、ASP、ASP.NET、PHP、JSP 等动态网页设计技术及.NET 框架。

Web 是一种基于浏览器/服务器、采用 Internet 网络协议的体系结构，是一种基于 Internet 的超文本信息系统。早期的 Web 页面是静态的，静态页面是用纯 HTML 代码编写的。后来，以 ASP、ASP.NET 和 Java 为代表的动态技术使 Web 从静态页面变成可执行的程序，从而产生了动态网页，大大提高了 Web 的动态性和交互性。ASP 是 Web 动态页面设计的基础，通过 ASP，Web 页面可以访问数据库，存取服务器的有关资源，使得 Web 页面具有强大的交互能力。Web 的交互性还表现在它的超链接上，因为通过超链接，使用户的浏览顺序和所到站点完全可由用户自行决定。

动态网页的实现一般采用客户端编程和服务器端编程两种程序设计方法。客户端编程就是客户端浏览器下载服务器中的程序来执行有关动态服务工作。常见的客户端编程技术有 VBScript、JavaScript、Java Applet 等。服务器端编程就是将程序员编写的代码保存在服务器中，当用户提出对某个网页的请求时，这个请求所要访问的页面代码都在服务器端执行，并把执行结果以 HTML 文件代码的形式传回浏览器。常见的服务器端编程技术有 PHP、JSP、ASP 和 ASP.NET。

Internet 是由各种不同类型、不同规模、独立管理和运行的主机或计算机网络组成的一个全球性特大网络。Internet 使用的网络协议是 TCP/IP 协议，凡是连入 Internet 的计算机都必须安装和运行 TCP/IP 协议软件。TCP/IP 协议是一个协议集，其应用层主要有：超文本传输协议 HTTP、远程登录协议 Telnet、文件传输协议 FTP 和域名服务系统 DNS 等。

IP 地址是识别 Internet 中主机及网络设备的唯一标识。但对大多数人来说，记住很多计算机的 IP 地址并不是一件容易的事，所以产生了域名服务系统 DNS，允许为主机分配字符名称，即域名。在网络通信时，由 DNS 自动实现域名与 IP 地址的转换。WWW 信息分布在全球，要找到所需信息就必须有一种说明该信息存放在哪台计算机的哪个路径下的定位信息。统一资源定位器 URL 是用来确定某信息位置的方法。

习　题　1

1.1　试简述 Web 的特点及应用。

1.2　试描述 Web 服务器向浏览器提供服务的基本过程。

1.3　请列举主要的动态网页设计技术。

1.4　TCP/IP 协议分成哪几个层次？每个层次的主要功能是什么？

1.5　请解释下列网络协议的作用：

Telnet　SMTP　FTP　DNS　HTTP　TCP　IP

1.6　名词解释：

域名　IP 地址　URL　Web　PHP　JSP　ASP　ASP.NET

1.7　.NET 框架由哪几部分组成？ASP.NET 与.NET 框架是什么关系？

第 2 章　Web 应用程序开发与运行环境

在第 1 章中，介绍了 Web 编程的基础知识。本章将对 Web 应用程序开发环境和常用工具做简单介绍，包括常用的工具软件、Dreamweaver MX 及 ASP.NET 应用程序的开发工具 Visual Studio.NET，这将为以后学习具体的编程方法和技术做好准备。

2.1　服务器端开发环境

服务器是对 Web 浏览器检索信息的请求做出响应，进而将 HTML 文档回传到客户机的浏览器屏幕上，或者运行服务器端程序的计算机。Web 的结构属于客户机/服务器系统，服务器端需要操作系统的支持，目前最常用的网络操作系统有 Windows NT、UNIX 和 Linux 等。

如果有条件和技术，可以在用户自己的本地计算机中构建 Web 服务器，如安装 Windows 系统的 IIS 组件，然后通过网络连接，为其他用户提供服务。但是如果没有条件和技术，可以在本地计算机中编写制作 Web 网页后，将其上传到托管或代理服务器中，利用专业服务器为用户构建的平台，完成 Web 网页的发布。现在许多网站可以提供发布空间服务，用户只要按照网站的使用要求，即可轻松完成 Web 网页的上传和发布。

服务器端的编程语言，除现在一般较少采用的 CGI 程序外，常用 ASP、Perl 和 PHP，还有微软公司近期推出的新一代 ASP.NET 语言，它直接与 Java 比拼，力图成为网络服务器端的标准语言。另外，如果仅用 Access 作为服务器端数据库，可以先在本地建立数据库，然后上传到服务器中，不必设定 ODBC 连接，直接用目录文件方式操作数据库，这样服务器中的工作就简化些。如果采用 SQL Server、Oracle 等其他类型的数据库作为服务器端数据库，就必须在服务器中创建数据库，还要进行 ODBC 的连接、设定等工作。

2.2　客户端开发环境

提起客户端编程语言，人们首先想到的是 HTML 语言，要制作 Web 网页就必须熟练掌握 HTML 代码。不过，随着功能强大的可视化网页制作工具的出现，使 Web 网页制作变得非常简单，生成的 Web 网页效果也更加丰富。用户可以将精力放在 Web 网页内容的组织与形式的创意上，而不必再放在如何编辑代码上。

虽然 HTML 在 Web 制作方面的地位有所下降，但是并不是说从事 Web 程序设计就不需要了解 HTML 语言了。如果用户希望自己成为高级 Web 程序设计人员，就必须学习 HTML 代码，应用 HTML 代码常常可以帮助用户解决一些棘手的问题。例如，当用户发现网页中有一些小错误时，如果通过启用 Dreamweaver 等软件进行修改，其麻烦程度是可想而知的，但是如果在记事本中直接查看 Web 网页的 HTML 代码，然后将出错的地方修改过来并保存，就非常快捷了。如果用户想编写脚本语言程序或编制复杂的 Web 网页（如一些特殊的页面控制），就必须了解 HTML 语言。

当然，要制作一个漂亮、直观的 Web 网页，除了要了解一些基本的 HTML 语言知识外，还需要掌握一种或几种工具。现在用于图形、图像设计及处理的工具很多，如 Photoshop、CorelDRAW、Fireworks MX、Freehand、Illustrator、PhotoExpress 等；Web 网页动画制作工具也很多，如 Flash MX、

Cool3D、3DS MAX R3、ImageReady 和 Fireworks MX 等；Web 网页制作工具主要有 Dreamweaver 及 FrontPage 等。

对于 Web 图形、图像设计来说，Photoshop 的图形处理能力比较强，Fireworks 的图像制作能力比较强，一般使用这两个软件基本可以胜任任何图形、图像创作的需要。对于动画制作而言，简单的动画使用 ImageReady 和 Fireworks 软件即可，不过复杂的动画最好使用 Flash 专业动画设计软件。

对于 Web 网页制作软件，常用 Dreamweaver 软件。Dreamweaver 提供图形设计界面，学起来都比较简单，生成的冗余代码少，站点管理、特效实现等轻而易举。值得提醒大家的是，Fireworks、Flash、Dreamweaver 这三个软件均由 Macromedia 公司推出，因此这三个软件在配合上比较完美。

2.3 网页设计工具 Dreamweaver MX

2.3.1 Dreamweaver MX 概览

Dreamweaver 是 Macromedia 公司开发的网页制作工具，它与 Macromedia 公司的另外两项产品 Firework 和 Flash 一起组成一套功能强大的网页创作系统，分别覆盖了网页制作、网页图形/图像处理和矢量动画这三个主要的网络创作领域。

做过网页的人也许最早接触的网页制作工具是微软公司的 FrontPage。它为可视化（所见即所得）的网页编辑提供了一个相当不错的编辑环境，但是较 Dreamweaver 而言，FrontPage 在 HTML 源代码的精确控制、易用性及对各种新技术的支持上都略逊一筹。这是人们在对网页制作工具进行评价时，更津津乐道于 Dreamweaver 的原因。

本书将介绍网页设计工具 Dreamweaver MX。它较以前的 Dreamweaver 版本，界面布局更简洁，面板排布更合理，功能更强大。为叙述简便，以下将 Dreamweaver MX 简称为 DW。

2.3.2 Dreamweaver MX 的特性

（1）精确性。DW 采用 Roundtrip HTML 技术实现对 HTML 源代码的精确控制，它能生成简洁高效的 HTML 代码。比如，在可视化编辑器中进行编辑时，可以在 HTML 源代码窗口中同步看到 HTML 源代码的变化；同样地，在 HTML 源代码窗口中直接编写代码时，也能马上在可视化编辑器中显示相应的可视化结果；甚至在可视化编辑器中可以对 HTML 标记直接进行选择、添加、修改或删除等操作。文档中如出现不配对的标记，将会用黄色显示提醒用户有错误需要修改。

（2）易用性。DW 的编辑界面相当友好，且操作简单。通过各种工具面板，可以非常方便地控制页面各种元素的属性。在不用手工输入一行代码的情况下，就可以制作出各种特效，比如动画、动态按钮、索引条、分层等。

（3）兼容性。兼容性是 DW 的一个非常优秀的特性，用它制作的页面能在各种浏览器上正确地显示。这在其他网页制作工具中是没有的，也是人们更倾向于它的一个最重要的原因之一。

2.3.3 Dreamweaver MX 界面介绍

启动 DW 之后，会看到如图 2-1 所示的界面。与 Windows 界面风格相比，DW 界面更相似于 PageMaker 和 Photoshop 等软件界面。它将各种操作和命令分布到不同的浮动面板上，可以随时激活和隐藏这些浮动面板，也正是这种灵活的窗口布局使得 DW 的操作更加便捷。

在图 2-1 启动界面中，可以从 4 个区域创建或调入文件。①单击中部的【创建新项目】，建立

相应类型的新文件。②单击中部的【从范例创建】，建立多种形式的新文件。③双击右部的【文件】列表中的文件名，调入指定目录中的文件。④单击左部的【打开最近项目】，可调入编辑过的文件。

图 2-1　Dreamweaver MX 启动界面

将 HTML 或其他文本格式的文件调入后，会以多页面的方式显示在文档窗口中，文件调入 DW 后的编辑界面如图 2-2 所示。

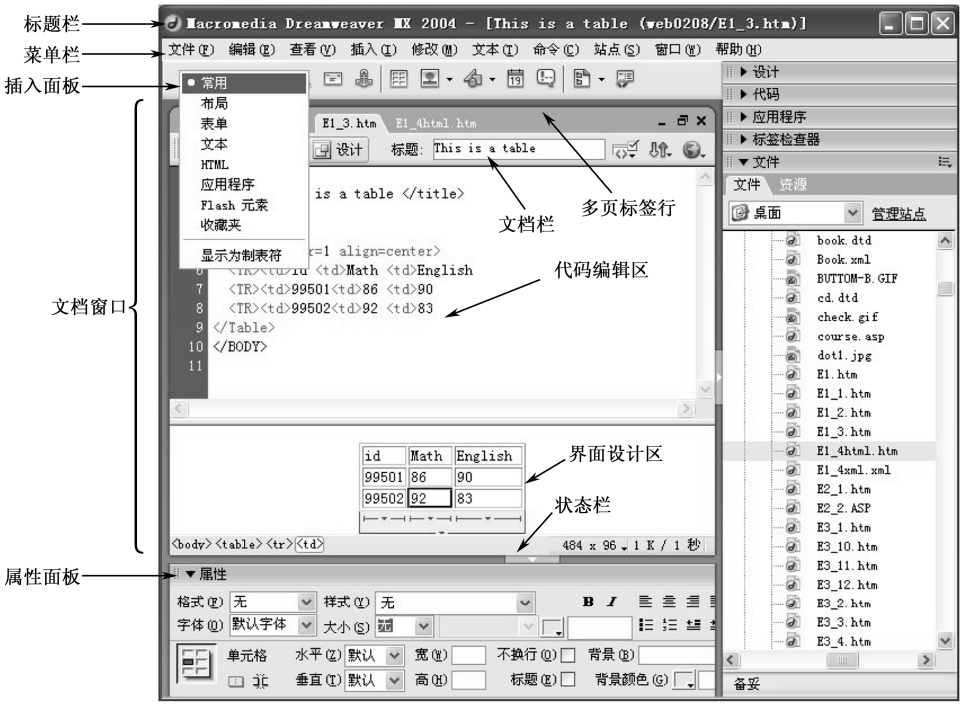


图 2-2　Dreamweaver MX 文件编辑界面

DW 的编辑环境从上到下依次为标题栏、菜单栏、插入面板、文档窗口、属性面板、文件面板等。在【窗口】菜单中单击【隐藏面板】可以显示或关闭所有的面板；单击下部和右部面板边框上的箭头按钮，可显示或关闭该组面板；插入面板等工具栏面板可以通过单击【查看】菜单的【工具栏】选项，显示或关闭该工具栏；工具栏也可由用户拖放到合适的位置。

1．标题栏

标题栏显示当前文档窗口中页面的标题，这个标题在 HTML 源文件中用<title>标记定义。标题后面的括号中是文件名，如果文件名后带有*号，则表示该文件包含尚未存盘的修改。

2．菜单栏

菜单栏分类罗列了所有的功能和命令，通过它可以完成所有的操作。

3．文档窗口

文档窗口是显示当前所编辑页面的窗口，如图 2-2 中间大块区域所示，它分为多页标签行、文档栏、代码编辑区、界面设计区及状态栏几部分。

（1）多页标签行。可以同时打开多个编辑页面，在层叠多窗口最大化时出现。

（2）文档栏。在这一行有三个按钮分别控制页面编辑的显示方式，分为单独的代码视图显示、拆分（代码编辑/设计页面上、下双显示，如图 2-2 所示）显示和单页面设计视图显示等方式。在本行还有标题文本修改框及浏览器预览图标等。

（3）代码编辑区。显示当前的 HTML 源代码，并按照颜色设置显示各种 HTML 语言标记。

（4）界面设计区。显示实际的输出效果，它是主操作区，显示当前页面的可视化结果。

（5）状态栏。显示页面的状态信息，它又分成三个部分。

① 标记选择器。选定文本或对象的标记将出现在文档窗口底部左边的标记选择器中。单击某一标记，可在文档窗口中高亮显示它的内容。单击<body>标记可选择文档的全部正文。如图 2-2 中，光标停在页面区的<td>格式的文字“ 92 ”上，显示该对象是在<body> <table> <tr> <td>标记之中。

② 窗口大小。单击这里将弹出窗口大小设置菜单，可以选择或设置所需要的窗口大小。或者拖动窗口边框动态地设置当前窗口的大小，宽和高的像素值就是页面显示的实际大小。

③ 文件大小/下载时间。显示的是估算的文档大小和页面的下载时间（包括所有独立文件，如图像和 Shockwave 动画）。下载时间默认以 56.0kb/s 估算，这可以兼顾编辑页面的效果和占时比。

4．插入面板

新版的 DW 将对象面板改为插入面板，图 2-2 所示为【常用】对象组的表格、图片等 10 个图标，该插入面板还有其他几组对象。

选择【窗口】菜单的【插入】项即可打开插入面板，它是各种网页对象的集合。DW 将常用对象分为几组，单击插入面板的向下箭头，会显示它们分别是【常用】、【布局】、【表单】、【文本】、【HTML】、【应用程序】、【Flash 元素】等。通过插入面板，可以快速地在网页中插入任何对象，如图片、表格、层、特殊字符等。下面简单介绍常用的每一个对象面板。

【常用】包含主页中最常用的一些对象，如图片、表格、超链接等。

【布局】包含常用的框架结构，如左/右分帧、上/下分帧等。

【表单】包含表单及所涉及的所有元素，如文本框、按钮、复选框、单选钮、列表框等。

【文本】包含一些特殊字符，如版权符号、注册商标符号、商标符号等。

【HTML】添加一些 Script 脚本等。

对象面板是 DW 中非常重要的一个面板，只有熟悉这些对象，才能随心所欲地在网页中添加各种对象。随着所编辑的文件类型的不同，插入面板会对应出现不同的对象组。

5．属性面板

选择【窗口】菜单中的【属性】项即可打开属性面板，它用来显示文档窗口中选定对象的各种属性。当单击页面中不同的对象时，自动出现的属性面板也有所不同。在属性面板中可以直接对各项属性值进行编辑和修改。

注意，属性面板一般分为上、下两块，在右下角有个打开/关闭下块的箭头，有些扩展属性安排在下块中。单击属性面板标题行的箭头，还可以收缩/展开该面板。

选定表格时，相对应的属性面板如图 2-2 下部所示。它显示了所选文字的格式、样式、字体、大小、颜色、对齐方式、链接（填写 URL 后，字体默认变为蓝色并带下画线，在页面显示时单击可跳转）、目标（为链接跳转时有效的 URL 指定显示页面的位置，_blank 为新窗口）等属性。由于该文本位于表格中，属性面板的下块显示了该表格的各种设置属性。

又如，选定一幅图片时，属性面板就变为显示图片的缩略图、大小、源文件名、对齐方式及其他属性等。

如图 2-3 所示，在 DW 编辑环境中新建了一个 HTML 文档，命名为 t1.htm，存到本机的目录中，如存于 D:\MyExp 目录中。这一步初学者要特别注意，新建文档后，要立即将文件存盘，其他需添加的对象，如图片文件等，都要复制到该目录中。

图 2-3　Dreamweaver MX 插入图片及热区编辑界面

首先插入图片。单击插入面板【常用】项的图片对象，在浏览窗口中选择确定，完成当前目录中图片文件的添加操作，这样在代码编辑区和界面设计区将自动插入<img>代码和图片的原样显示，在页面的光标位置就插入了图片。

然后进行热区编辑操作。单击设计区的图片，有方框标出已选定该图片，然后在下方的属性区的左下角选择矩形热区，用鼠标在图片的指定区域单击定位后再拖拽出一个矩形区域，即为“ 图片热区 ”，相对应的<map>代码会自动生成；再单击该图片，对该热区的链接属性进行设定。在下方属性区【链接】文本框内填写要指向的某个 URL，以后该网页在浏览器窗口内显示时，鼠标移到该矩形区域时，鼠标指针将变为手形，单击它就可以完成超链接的跳转。

如图 2-4 所示，要插入一个表格，先单击插入面板【常用】项的表格对象，在设定了表格的行/列值、宽度、边框粗细、边距、间距等属性值后，在代码编辑区和界面设计区中将自动插入<table>代码和表格的实际显示，在“所见即所得”的页面上，拖动边框线可改变表格的大小，选中多个单元格后，再单击属性区左下角的 合并按钮可将这些单元格合并为一个单元格。

如图 2-5 所示，要插入一幅 Flash 图像，先单击插入面板【常用】项的媒体对象，选择【Flash】，再选择当前目录中的.swf 文件，在代码编辑区和界面设计区中将自动插入<object>代码和图片的轮廓显示，这样在页面的光标位置就插入了 Flash 图像。

图 2-4　Dreamweaver MX 插入表格编辑界面

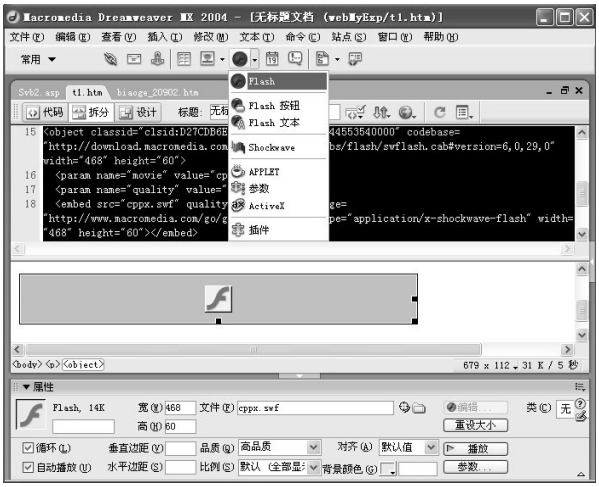


图 2-5　Dreamweaver MX 插入 Flash 图像编辑界面

如图 2-6 所示，要插入 Frame 框架，先单击插入面板【布局】项的框架对象，选择【左侧和顶部】对象，即页面分为左、右上、右下三个独立的显示区，DW 会自动生成 4 个.htm 文件。在各个显示区域中单击，多页标签行的同一个标签上出现该文件名，代码编辑区出现对应的代码；各个文件要分别独立存盘，如 left.htm、top.htm、t1.htm；当光标移至窗体边框或框架分隔处时，鼠标指针变为横或竖向箭头，单击鼠标，代码编辑区显示所设定框架布局的第 4 个.htm 文件的代码，如存盘为 index.htm 文件，则该文件包含自动生成的<frame>代码和布局的三个<src>指向的文件。在属性面板中也可修改框架属性和各区域大小等。

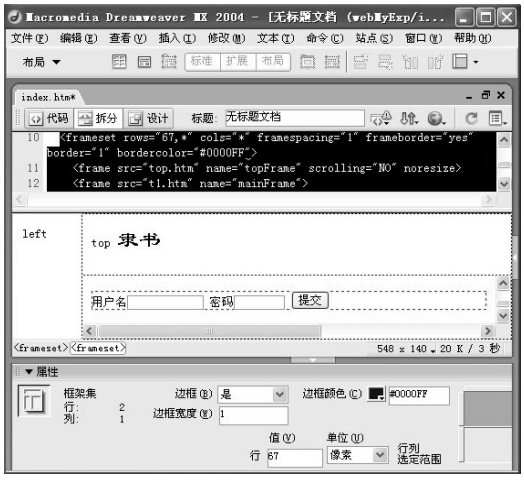


图 2-6　Dreamweaver MX 插入 Frame 框架和 Form 表单编辑界面

在【窗口】菜单中，单击【框架】项，将在此行前加√，在 DW 编辑界面右部会显示【框架】

面板，在此面板窗口内，标出了框架的形状和框架的几个显示区域。

在图 2-6 的下方插入了一个 Form 表单，在页面上设定好光标位置，再单击插入面板【表单】项的表单对象，出现红色虚线框，再单击【表单】项的文本字段对象、按钮对象，添加文字，页面上将出现如图 2-6 所示的 Form 表单，代码编辑区显示相应的代码。

在以上 DW 的简单介绍及基本应用的例子中, 我们已初步看到了其强大的功能和方便的操作, 开发者可以不用写一行代码，仅用鼠标操作，就可以制作出精彩的网页。

2.4　Visual Studio.NET 开发工具

Visual Studio. NET 是 Microsoft 公司于 2002 年正式推出的一个集成开发环境，它集源程序编辑、编译、链接及项目管理和程序发布于一体，是开发 ASP.NET 应用程序强大的工具。尽管不使用 Visual Studio.NET 也可以开发出复杂的应用程序，但使用它无疑会更高效。用它可以创建 Windows 平台下的 Windows 应用程序和 Web 应用程序，或者创建网络服务、智能设备应用程序和 Office 插件。Microsoft 公司先后推出过 2002、2003、2005、2008 等多个版本，目前的最新版是 Visual Studio 2013。本节主要介绍 Visual Studio 2012。

Visual Studio 2012 提供了改进的语言和数据功能，编程人员可以利用这些功能更轻松地构建解决方案。开发人员还能够在同一开发环境内创建面向多个.NET Framework 版本的应用程序。对于 Web 开发，Visual Studio 2012 也提供了新的模板、更优秀的发布工具和对新标准（如 HTML5 和 CSS3）的全面支持。使用 Visual Studio 2012 集成开发环境，IIS 不再是开发 ASP.NET Web 应用程序的必要条件，因为 Visual Studio 2012 自身就搭载了一种本地 Web 服务器。

2.4.1　Visual Studio 2012 的安装

1．系统配置要求

运行 Visual Studio 2012 软件需要一定的硬件和软件支持，所以安装之前必须先检查计算机的软、硬件配置是否满足安装要求。具体要求如下。

（1）操作系统

Microsoft Windows 7；Microsoft Windows Server 2008 R2；Windows 8。

（2）硬件配置

最低要求：1.6 GHz CPU、1GB RAM、1024×768 显示器、5400 rpm 硬盘。

建议配置：2.2 GHz 或速度更快的 CPU、2GB 或更大容量的 RAM、1280×1024 显示器、7200 rpm 或更高转速的硬盘。

以上是它的自述文档中给出的最低配置。如果可能, 配置越高越好, 这样才可以充分发挥.NET 的优势，提高工作效率。

另外, 还需准备足够的磁盘空间。建议安装分区至少有 20GB 的可用空间, 因为将 Visual Studio 2012 完全安装在系统分区（推荐）, 至少需要占用 5GB 的空间, 若包括 MSDN 的安装需要至少 13GB 的空间。

2．安装 Visual Studio 2012

以 Visual Studio 2012 Ultimate 版本安装为例，具体步骤如下。

（1）将 Visual Studio 2012 安装光盘放入光盘驱动器，双击其中的 vs_ultimate.exe 可执行文件即可开始安装，安装最好使用管理员方式运行该程序。运行后进入如图 2-7 所示的 Visual Studio

2012 安装程序界面。

（2）选择安装目录，需要 8GB 的空间，然后勾选同意许可条款，完成之后，单击“下一步”按钮继续安装。在如图 2-8 所示的要安装的可选功能界面中，最好全部勾选。设置好以后，单击“安装”按钮，这时进入真正的安装阶段。一段时间后，系统显示安装成功。

图 2-7　Visual Studio 2012 安装程序界面

图 2-8　Visual Studio 2012 安装功能选项

2.4.2　Visual Studio 2012 集成开发环境

Visual Studio 2012 集成开发环境（IDE）与 Microsoft 公司的其他应用程序界面类似，它由以下若干界面元素组成：标题栏、菜单栏、工具栏、工具箱、文档窗口、解决方案资源管理器窗口和属性窗口等，如图 2-9 所示。

图 2-9　Visual Studio 2012 集成开发环境

1．标题栏

标题栏位于窗口顶端，显示网站或项目名称以及系统的工作模式。启动 Visual Studio 2012 后，标题栏显示的是当前运行的网站或项目，此时处于设计状态。随着工作方式的变化，标题栏显示的信息也会随之发生变化。Visual Studio 2012 有以下三种工作模式。

设计模式：此时可进行用户界面设计、属性设置和代码编写。

运行模式：程序处于运行状态，可以查看程序的运行结果。

中断模式：程序运行暂时中断。单击“ 继续 ”按钮，程序继续执行。

2．菜单栏

菜单栏显示所有可用的命令。通过鼠标单击或通过 Alt 键加上菜单项上的字母执行菜单命令。

3．工具栏

为了操作更方便、快捷，菜单项中常用的命令按功能分组分别放入相应的工具栏中。通过工具栏可以迅速地访问常用的菜单命令。常用的工具栏有标准工具栏和调试工具栏。

4．工具箱

它是 Visual Studio 2012 的重要工具，通常位于窗口的左侧。它提供 Windows 窗体应用程序开发所必需的控件，主要包括 HTML 控件和 Web 服务器控件等多组控件。当需要某个控件时，双击所需的控件直接将控件加载到设计窗体上，或先单击选择需要的控件，再将其拖动到设计窗体上。

5．属性窗口

属性窗口的作用是显示和设置选定控件的属性值。如图 2-10 所示，其左侧是属性名称，右侧是属性值。除此之外，属性窗口还可以管理控件的事件，方便编程时对事件的处理。属性窗口采用“ 按分类顺序 ”和“ 按字母顺序 ”两种方式管理属性和事件。

6．解决方案资源管理器窗口

解决方案资源管理器窗口主要用于代码查看、“ 设计 ”视图与“ 源 ”视图的切换等。它以树形结构进行项目文件的组织和管理，如图 2-11 所示。

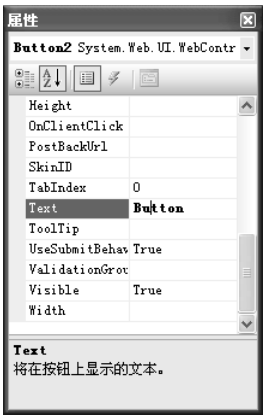


图 2-10　属性窗口

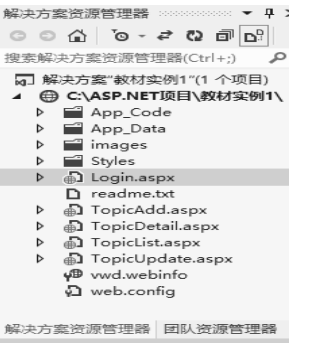


图 2-11　解决方案资源管理器窗口

7．文档窗口

文档窗口是用户进行界面设计和代码编辑的场所，窗口中显示的是正在处理的文档。单击“ 视图 ”选项卡可以实现“ 设计 ”视图和“ 源 ”视图之间的切换。

8.“视图”选项卡

“ 视图 ”选项卡用于选择同一文档的不同视图。“ 设计 ”视图近似 WYSIWYG（所见即所得）的编辑画面，允许在用户界面或网页上摆放控件。“ 源 ”视图是页的 HTML 编辑器，用于显示文件或文档的源代码。“ 拆分 ”视图将同时显示文档的“ 设计 ”视图和“ 源 ”视图。图 2-9 是“ 拆分 ”视图。

2.4.3 Visual Studio 2012 集成开发环境的使用

用 Visual Studio 2012 创建 ASP.NET 应用程序，可以分为以下 5 个主要步骤：

（1）创建一个网站；

（2）在网站中添加一个空白的 ASP.NET 应用程序页面文件；

（3）设计应用程序界面；

（4）编写应用程序的事件代码；

（5）调试和运行应用程序。

下面以一个简单的例子来演示 ASP.NET 应用程序的建立过程，使读者能在较短的时间内掌握 Visual Studio 2012 集成开发环境的使用方法。

【例 2-1】设计一个如图 2-12 所示的登录界面程序。单击“ 登录 ”按钮，若输入密码为“ 123 ”，将显示“ 你已成功登录！”，如图 2-13 所示；否则，显示“ 登录失败！”。

图 2-12 程序登录界面

图 2-13 程序运行结果

（1）新建网站

选择“ 文件 ”菜单的“ 新建网站 ”命令，在下级菜单中，单击“ 网站 ”命令，出现如图 2-14 所示的“ 新建网站 ”对话框。选中“ ASP.NET 空网站 ”模板，在“ Web 位置 ”框中选择“ 文件系统 ”。单击“ 浏览 ”按钮，输入要保存网站的网页文件夹名（如 D:\我的文档\Visual Studio 2012\WebSites\WebSite1）。单击“ 模板 ”列表中“ Visual C# ”项，最后单击“ 确定 ”按钮。VS.NET 将创建一个名为 WebSite1 的新网站，同时自动创建一个名为“ default.aspx ”的主页面文件。默认以“ 源 ”视图方式显示该页，在该视图下可以查看到页面的 HTML 元素。

若网站中还需要其他页面文件，可由设计者自行添加。向网站中加入页面文件的方法是：在“ 解决方案管理器 ”窗口的该网站名上右击鼠标，弹出一个快捷菜单，选择“ 添加 ”→“ 添加新项 ”命令，出现如图 2-15 所示的“ 添加新项 ”对话框，选中“ Web 窗体 ”模板，输入新页面文件名 Login.aspx。在“ 语言 ”列表中，选择希望使用的编程语言“ Visual C# ”，勾选“ 将代码放在单独的文件中 ”复选框，这样就创建了一个代码和 HTML 分离的 Web 窗体。

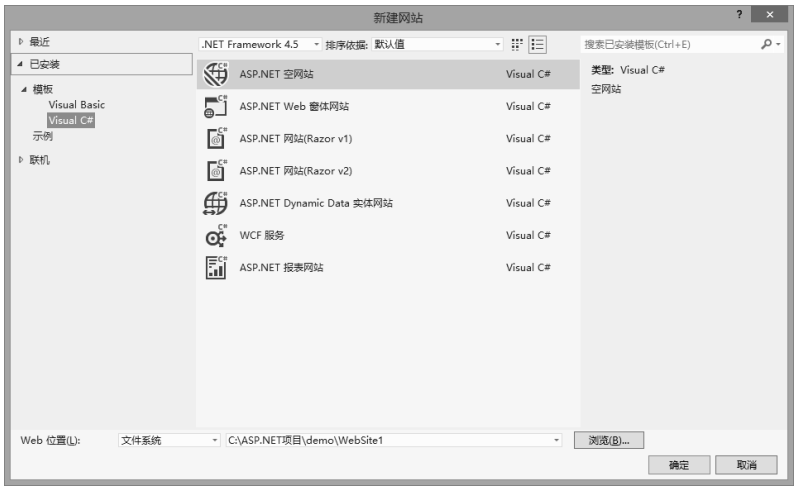


图 2-14　“ 新建网站 ” 对话框

图 2-15　“ 添加新项 ” 对话框

（2）利用工具箱中的相关控件设计应用程序界面

单击 “ 视图 ” 选项卡切换到 “ 设计 ” 视图。在工具箱中单击 “ 标准 ” 类别。根据图 2-12 所示的界面，在文档窗口中放置一个 3 行 1 列的 HTML 表格，在表格中合适的位置分别输入文本 “ *登录名 :” 和 “ *密码 :”，同时放置两个 Textbox 控件（TextBoxLoginName 和 TextBoxPassword）和一个 Button 控件（ButtonLogin），选定 ButtonLogin 控件，修改 “ Text ” 属性值为 “ 登录 ”。

（3）编写程序代码

在 “ 设计 ” 视图中，双击 ButtonLogin 控件，自动创建该按钮的单击事件处理程序 “ ButtonLogin_Click ”，并切换到 “ 源 ” 视图，如图 2-16 所示。

```
protected void ButtonLogin_Click(object sender, EventArgs e)
    {


    }
```

在程序 “ ButtonLogin_Click ” 中添加以下程序代码：

```
if  (TextBoxPassword.Text=="123" )
  {
    Response.Write("你已成功登录！");
```

```
        }
         else
        {
            Response.Write("登录失败！");
        }
```

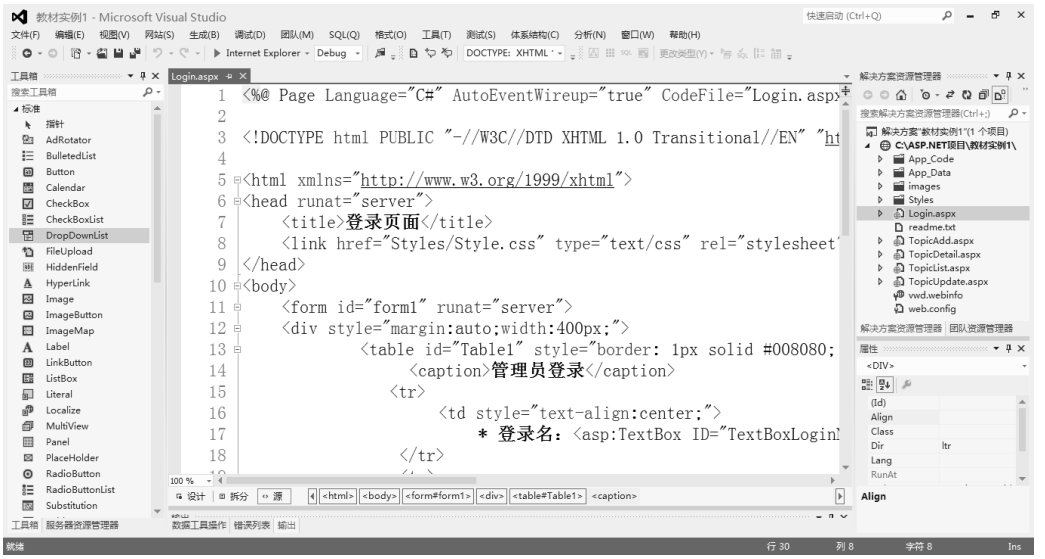


图 2-16 “ 源 ” 视图

（4）运行程序

首先，在解决方案管理器窗口选中待执行的程序 Login，单击鼠标右键，在快捷菜单中单击【设为起始页】命令。然后单击工具栏上的“ 启动调试 ” 按钮，即可在浏览器中查看运行结果。输入用户名“ admin ”，输入密码 “ 123 ”，运行结果如图 2-13 所示。

（5）保存文件

运行程序前最好先保存程序，这样可以避免由于意外发生而丢失设计好的程序。选择【文件】菜单中的【全部保存】命令，可将所有文件保存到相应的文件夹里。

（6）退出集成开发环境

单击“ 文件 ” 菜单中的“ 退出 ” 命令。

本 章 小 结

本章简单介绍了 Web 程序开发环境 Dreamweaver MX，它是编制 HTML 文件和 ASP 文件非常实用的集成开发环境。Dreamweaver MX 采用 Roundtrip HTML 技术实现对 HTML 源代码的精确控制，能生成简洁、高效的 HTML 代码；它编辑界面友好，且操作简单。此外，它的一个非常优秀的特性是兼容性，使用它制作出的页面能在各种浏览器上正确地显示。Visual Studio.NET 是 Microsoft 公司推出的一个集成开发环境，它集源程序编辑、编译、链接及项目管理和程序发布于一体，是开发 ASP.NET 应用程序的强大工具。尽管不使用 Visual Studio.NET 也可以开发出复杂的应用程序，但使用 Visual Studio. NET 无疑会更加高效。使用它可以创建 Windows 平台下的 Windows 应用程序和 Web 应用程序，或者创建网络服务、智能设备应用程序和 Office 插件。

习　题　2

2.1　常用的服务器端编程语言有哪些？

2.2　常用的 Web 网页制作工具和 Web 网页动画制作工具有哪些？

2.3　目前使用什么工具开发 ASP.NET 应用程序？

上机实验 2

实验 2.1　Dreamweaver MX 的使用。

【目的】学会使用 Dreamweaver MX。

【内容】用 Dreamweaver MX 制作如图 2-17 所示的个人简历表格。

姓名	马卫	出生日期	1983.06.04	性别	男	籍贯	江苏.东台
学历	本科	毕业院校	盐城师范学院		专业	计算机科学与技术	
家庭地址	江苏省东台市富安镇潘陈村一组5号						
简历	东台是一个很美的小城，我生活在这里，学习在这里， 但愿南师大是我人生的一次飞越！一次腾飞！						
通讯	• 电话：025-5893804 • Email:maweitian@163.com • QQ:8928253						
备注	• 个人主页：http://home.yctc.edu.cn/mawei • 本班同学名单：Namelist.htm						

图 2-17　用 Dreamweaver MX 制作的个人简历表格

【步骤】

（1）打开 Dreamweaver MX。

（2）在【插入面板】的【常用】对象组中单击【表格】图标，生成 6 行 8 列的表格。

（3）调整表格各栏的高度和宽度，部分单元格需要合并。

（4）在单元格中设置字体、字号、字形、颜色、大小等，根据需要添加超链接和列表标记点。

（5）插入照片或其他图片，调整图片大小，固定约束比及高宽。

（6）编制好如图 2-17 所示表格后，另存为 ex2-1.htm 文件。

*（7）选做：在表格上方添加如图 2-3 所示的 banner 图，带图片热区。

*（8）选做：在表格下方添加如图 2-5 所示的.swf 图——Flash 动画图。

（9）双击 ex2-1.htm 文件，在浏览器中观察其显示效果。

实验 2.2　Visual Studio 2012 的安装与使用。

【目的】学会使用 Visual Studio 2012。

【内容】Visual Studio 2012 的安装与使用。

【步骤】

（1）按照本章 2.4.1 节的介绍，练习 Visual Studio 2012 的安装过程。

(2)按照本章 2.4.3 节例 2-1 的介绍, 练习 ASP.NET 应用程序的建立过程, 从而掌握 Visual Studio 2012 集成开发环境的使用方法。

第 3 章　HTML 与 XML

HTML 和 XML 是两种重要的 Web 基础语言，本章将讲述它们的基本语法和应用。

超文本标记语言 HTML（Hypertext Markup Language）是在万维网上建立超文本文件的语言，它是万维网的核心计算机语言。创建 Web 站点时，需使用 HTML 语言组织 Web 页面放置文本、图像、音视频等多媒体信息内容，以及表单和超链接等可以进行交互的内容。可扩展标记语言 XML（eXtensible Markup Language）是万维网联盟（World Wide Web Consortium，W3C）于 1998 年 2 月发布的标准，目前已成为互联网标准的重要组成部分。

3.1　页面设计概述

一般来说，Web 网站开发的全过程大致分为 5 个阶段：策划与定义、设计、开发、测试和发布。首先要根据建站目的和定位进行策划与定义，确定网站风格、栏目、布局方式等；接下来要进行页面设计和后台程序开发。其中页面设计包括静态页面设计和动态页面设计，本章所要讨论的就是静态页面设计所涉及的有关技术，揭示相关页面设计技术的本质。

静态页面设计技术主要采用 HTML 或 XML 来完成。对于静态页面，用户只能浏览 Web 服务器上预先安排好的信息。设计静态页面主要使用各种页面开发工具，如 Dreamweaver 等。

HTML 语言自 1993 年 6 月由互联网工程工作小组（IETF）作为工作草案发布以来，已先后推出了 HTML2.0、HTML3.0、HTML3.2、HTML4.0、XHTML 及 HTML5 等多个版本。其中 HTML3.0 和 HTML4.0 规范对于网页设计尤为重要。HTML3.0 提供了很多新特性，如表格、文字绕排和复杂数学元素的显示等。1997 年 12 月推出的 HTML4.0 将 HTML 语言推向一个新高度，该版本倡导两个理念：一是将文档结构和显示样式分离；二是更广泛的文档兼容性。HTML4.0 对以前版本的标记进行了扩充，它将页面中的文字、图像等都作为对象来处理，并可通过脚本语言对其特性和变化予以控制。由于同期层叠样式表（CSS）的配套推出，使得 HTML + CSS 的网页制作能力达到了新的高度。1999 年 12 月，W3C 网络标准化组织推出改进版的 HTML4.01，该语言成熟可靠，一直沿用至今。

2000 年底，W3C 组织公布发行了 XHTML1.0 版本。XHTML 是一种增强了的 HTML，它的可扩展性和灵活性将满足未来网络应用更多的需求。虽然 XML 的数据转换能力强大，完全可以替代 HTML，但面对成千上万已有的基于 HTML 语言设计的网站，直接采用 XML 还为时过早。因此，在 HTML4.0 的基础上，用 XML 的规则对其进行扩展，就得到了 XHTML。所以，建立 XHTML 的目的是实现 HTML 向 XML 的过渡。所以，本质上说，XHTML 是一个过渡技术，结合了 XML 的强大功能及 HTML 的简单特性。

为了推动 Web 标准化运动的发展，一些公司联合起来，成立了 Web 超文本应用技术工作组（Web Hypertext Application Technology Working Group，WHATWG）。WHATWG 致力于 Web 表单和应用程序，而 W3C（World Wide Web Consortium，万维网联盟）专注于 XHTML2.0。2006 年，双方决定进行合作创建一个新版本 HTML，即后来的 HTML5。HTML5 草案的前身名为 Web Applications1.0，于 2004 年被 WHATWG 提出，2007 年被 W3C 接纳。历经 8 年艰辛努力，于 2014 年 10 月，万维网联盟终于宣布 HTML 5 标准规范制定完成。HTML 5 是一个突破性的版本，旨在

实现跨平台和更强的媒体支持。HTML 具有良好的标准性、多设备跨平台、自适应网页设计、提高可用性和改进用户友好体验、更多的多媒体元素支持、对站点优化的更好支撑等优点。目前已有 Chrome、IE、Firefox、Opera、360、搜狗、傲游及 QQ 等浏览器的高版本支持 HTML5。

3.2　超文本标记语言 HTML

HTML 源于“ 标准通用标记语言 ”（Standard Generalize Markup Language，SGML）的设计概念。SGML 的目的是为了使网络上文档格式统一，易于交流。SGML 采用“ 标记 ”进行描述。SGML 标记，英文称为 tag，就是在文档需要的地方，插入特定记号，来控制文档内容的显示，这就是文档格式定义。HTML 采用 SGML 的“ 文档格式定义 ”概念，通过标记与属性对一段文本的语义进行描述，并提供由一个文件到另一个文件或在一个文件内部不同部分之间的链接。HTML 标记是区分文本各个部分的分界符，用于将 HTML 文档划分成不同的逻辑部分（如段落、标题等），它描述文档的结构，与属性一起向浏览器提供该文档的格式化信息以传递文档的外观特征。

HTML 是一种文本标记语言，而非编程语言。HTML 文件是普通文本文件，与平台无关，可用任何文本编辑器进行编辑，通常文件扩展名为.htm 或.html。

3.2.1　HTML 文档结构

1．一个示例——创建《Web 程序设计》课程网站主页面

为使读者对 HTML 文件有一个整体了解，先看一个 HTML 文件示例。

【例 3-1】“《Web 程序设计》课程网站 ”主页面，如图 3-1 所示。

图 3-1　“《Web 程序设计》课程网站 ”主页面

该网页以表格作为页面的总布局方式，页面设计中使用了常用的 HTML 标记，包括：表格、表单、文字显示控制、加入图片、超链接、水平线、换行、分段、设置页面背景图片等。我们用记事本打开该页面对应的 HTML 文档，其内容如下：

```
<Html>
    <Head>
    <Meta Http-equiv="Content-Type" Content="text/html; Charset=gb2312">
    <Title>欢迎访问《Web 程序设计》课程网站</Title></Head>
```

文档头部标记

```
 <Body Background="bg.jpg">
<Div Align="center">
<Font size=7 Color=red Face="隶书">
《Web 程序设计》课程网站</Font>
</Div>
<Hr Color=blue Border="2"><Br>
<Table Width="497" Border="0" Align="center">
<TR>
   <Th Width="120" Scope="col"><Br>
   <Strong><A Href="home.htm">本站首页</A></Strong></Th>
   <Th Width="120" Scope="col"><Br>
   <Strong><A Href="jxdg.aspx">教学大纲</A></Strong></Th>
   <Th Width="120" Scope="col"><Br>
   <Strong><A Href="jxnr.aspx">教学内容</A></Strong></Th>
   <Th Width="120" Scope="col"><Br>
   <Strong><A Href="zxjl.aspx">在线交流</A></Strong></Th>
   <Th Width="120" Scope="col"><Br>
   <Strong><A Href="cslx.aspx">测试练习</A></Strong></Th>
</TR>
</Table>
<Br><Br>
 <Form>
        <Center>请输入用户名和密码：<Br><Br>
          用户名：<Input Type=text Name="xm" Size=10 Value=""><Br><Br>
          <Img Src="passwd.gif" Width="20" Height="20">密码：
     <Input Type=password Name="kl" Size=10 Value=""><Br><Br>
          <Input Type=submit Name="ok" Value="提交">
        </Center>
 </Form>

 <!-- 表单结束 -- >
</Body>
</Html>
```

体部起始标记，并设置页面背景图片

分段，设置标题文字格式

加入水平线，换行

表格，显示系统功能选项

输入用户名和口令表单

注释

体部结束标记和文档结束标记

从例 3-1 我们看到，HTML 文档是一个文本文件，其中包含 HTML 标记和属性形式的指令。双击 HTML 文件名即可在浏览器中显示页面内容。

HTML 标记用一对<>中间包含若干字符表示，通常成对出现，前一个是起始标记，后一个为结束标记，如<Html>…</Html>、<Head>…</Head>等。但也有部分标记非成对出现，如例 3-1 中出现的换行标记
。HTML 标记是大小写不敏感的。大部分标记都带有一个或多个属性，其中标记名告诉浏览器标记的用途，而属性（如果有的话）则为浏览器提供执行标记命令所需的附加信息。例如：

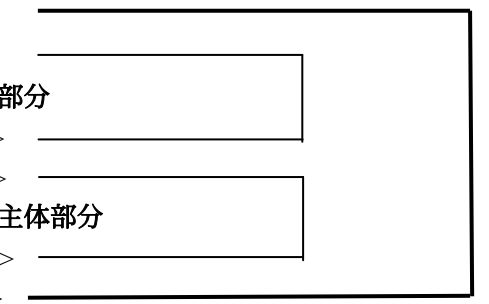
```
<Font size=7 Color=red Face="隶书">《Web 程序设计》课程网站</Font>
```

其中，Font 是标记名，告诉浏览器设置由<Font>及</Font>所界定的文字显示属性，而 Color 和 Face 为属性，用于设置文字的颜色和字体。

有些标记（如例 3-1 中的 Body）还包括一些事件，通过设置事件代码，当该事件产生时，事件代码便被执行。事件代码用脚本语言编写，目前常用的脚本语言为 JavaScript 和 VBScript。脚本语言编写的程序用 Script 标记括起来，Language 属性告知浏览器 Script 标记括起的脚本是用什么脚本语言编写的。例如，用 JavaScript 脚本语言，则设置 Language="JavaScript"或 Language="JScript"。

2．HTML 文档的基本构成

HTML 文档的基本结构如下：



HTML 页面以<html>标记开始，以</html>结束。在它们之间，是头部和体部。头部用<head>…</head>标记界定，一般包含网页标题，以及文档属性参数等不在页面上显示的网页元素。体部是网页的主体，内容均会反映在页面上，用<body>…</body>标记来界定，页面的内容组织在其中。页面的内容主要包括文字、图像、动画、超链接等。

3.2.2　HTML 基本标记

1．基本 HTML 标记

HTML 标记限定了文档的显示格式，分为头部和体部标记。

（1）头部标记

<head>，</head>　　//HTML 文件头部起始和结束标记。

<title>，</title>　　//HTML 文件的标题，是显示于浏览器标题栏的字符串。

<style>，</style>　　//CSS 样式定义，详见第 4 章。

<meta>　　//“元”标记，位于<head>与<title>标记之间，提供网页信息。

其中，<meta>标记主要用来为搜索引擎提供页面主题相关信息，包括 HTTP 标题信息（http-equiv）和页面描述信息（name）。<meta>标记的三种主要属性如下。

① name：meta 名字，描述网页，与 content 配合使用说明网页内容，便于搜索引擎进行查找和分类。

② http-equiv：说明 content 属性内容的类别。

③ content：定义页面内容，一些特定内容要与 http-equiv 属性配合使用。

name 与 content 属性配合使用的部分含义如下。

① name="keywords"，则 content 为搜索引擎提供的关键字列表。

② name="description"，则 content 与页面内容相关。

③ name="author"，则 content 为作者信息。

④ name="copyright"，则 content 为版权信息。

http-equiv 与 content 属性配合使用的部分含义如下。

① http-equiv="content-Type"，则 content 中是页面使用的字符集。

② http-equiv="content-language"，则 content 中是页面语言。

③ http-equiv="refresh"，则 content 中是页面刷新的时间。

④ http-equiv="expires"，则 content 中是页面在缓存中过期的日期。

例如：

```
<meta name="keywords" content="news, flood">
//设定关键字为 news, flood
<meta name="description " content="关于洪涝灾害的新闻">
//设定网页描述为"关于洪涝灾害的新闻"
<meta http-equiv="content-Type" content="text/html; Charset=gb2312">
//设定网页所使用的字符集为 GB2312，即汉字国标码
<meta http-equiv="content-language" content="zh-CN">
//设定网页所使用的语言
<meta http-equiv="expires" content="Aug,30,2015 00:00:00 GMT">
//设定网页在缓存中过期的日期为 2015-8-30，一旦过期，需要到服务器上重新下载
```

（2）体部标记

基本的体部标记包括 body、文字显示和段落控制标记、设置图像和超链接、列表和预定义格式标记等，用来在网页中插入文本、表格、超链接、多媒体等各类对象，进行排版等。

1）<body>和</body>标记

表明 HTML 文件体部的开始和结束，body 标记属性及含义列于表 3-1 中。

表 3-1　body 标记属性表

属 性 名	取　值	含　义	默 认 值
bgcolor	颜色值	页面背景颜色	#FFFFFF
text	颜色值	文字的颜色	#000000
link	颜色值	待链接的超链接对象的颜色	
alink	颜色值	链接中的超链接对象的颜色	
vlink	颜色值	已链接的超链接对象的颜色	
background	图像文件名	页面的背景图像	无
topmargin	整数	页面显示区距窗口上边框的距离，以像素点为单位	0
leftmargin	整数	页面显示区距窗口左边框的距离，以像素点为单位	0

例如：

```
<body topmargin=5 background="images/back057.gif" text="#ff0000" link="yellow" vlink="#00ff00">
```

HTML 文件中许多标记都有颜色控制，颜色值在 HTML 中有如下两种表示方法。

① RGB 值表示。用颜色的十六进制 RGB 值表示，形如"#RRGGBB"。如"#ff0000"表示红色，"#0000ff"表示蓝色。

② 英文单词表示。如"red"表示红色，"blue"表示蓝色。

2）文字显示和段落控制标记

文字显示属性主要有字体、字号、颜色，段落控制显示对象的分段。常用的文字显示和段落控制标记列于表 3-2 中。

表 3-2　常用的文字显示和段落控制标记表

标 记 名	含　义
,	以属性 face、size、color 控制字体、字号、字颜色的显示特性
<I>,</I>	斜体
<B>,</B>	粗体
<U>,</U>	加下画线
_,	下标
[,]	上标
<big>,</big>	大字体
,	小字体

续表

标　记　名	含　　义
<h1>—<h6>	标题格式，数字越大，显示的标题字越小
,	分段标记，属性有 align：left（左对齐）、center（居中对齐）、right（右对齐）
,	块容器标记，其中的内容是一个独立段落
<hr>	分隔线，属性有：width（线的宽度）、color（线的颜色）
,	居中显示

【例 3-2】一个包含文字显示和段落控制标记的 HTML 文件示例。

```
<html><head><title>文字显示和段落控制</title></head>
<body background="images/back057.gif" text="#ff2222">
<center><h1>一级标题</h1></center><hr width=90% color=green>
<font face="黑体" size=7 color="0000ff">这是黑体，大小为 7 号字，蓝色</font><br>
<p>这是一个段落<br>
<I>这是斜体</I><B>这是粗体</B><U>这是下画线字体</U>
<big>这是大字体</big><small>这是小字体</small>
这是下标字体<sub>1</sub>这是上标字体<sup>2</sup><br>
<font face="楷体" size=6 color="cc8888">
<I><B><U>这些标记还可以混合使用</U></B></I></font></p>
<p align=center>这是另一个段落<br>
<B>    以下是转义序列</B><br>
&lt; 小于号；&gt; 大于号；&amp; 与号；&quot; 双引号；例如:a&gt;b</p>
</body></html>
```

例 3-2 在浏览器中显示的效果如图 3-2 所示。

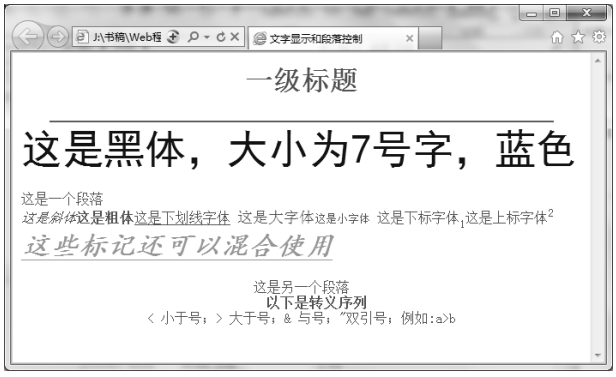


图 3-2　文字显示和段落控制

本例有两点需要说明。

① 转义序列。在 HTML 文件中有些符号有特殊用途，如“ < ”、“ > ”等，它们相当于高级语言的关键字, 如果在 HTML 的正文中需要显示它们, 就必须用转义序列。最常用的转义序列是 ——空格，如果 HTML 需要显示一个或多个空格，必须用该转义序列。其他常用的转义序列还有：<——小于号；>——大于号；&——与号；"——双引号。例如要显示 x<y, 在 HTML 文件中需要书写为 x<y。

② 换行标记
。在 HTML 中输入的硬回车符并不引起浏览器显示换行，要使浏览器在指定处换行，要用标记
。
是非成对标记。

3）图像标记

主要有以下几种图像格式可以被浏览器解释：GIF 格式（.gif 文件）、X 位图格式（.xbm 文件）、JPEG 格式（.jpg、.jpeg 文件）及 PNG 格式（.png 文件）。

在例 3-1 中我们已经看到，用图像标记<img>可以在页面中插入一幅图像。例如：

```
<img src="images/a.gif" border=0 width=200 height=100 alt="示例">
```

<img>标记包括如下属性。

① src：指明图像文件的地址，该属性值必须指明。该值可以是一个本地文件名或一个 URL 形式，如 http://member.shangdu.net/images/logo.gif。

② border：指明图像边框的粗细，值为整数。若为 0，则表示无边框；值越大，边框越粗。

③ width：图像宽度，值为整数，单位为屏幕像素点数。若不指出该属性值，则浏览器默认按图像的实际尺寸显示。

④ height：图像高度，值为整数，单位为屏幕像素点数。若不指出该属性值，则浏览器默认按图像的实际尺寸显示。

⑤ alt：当鼠标指针移至该图像区域时，将以一个小标签显示该属性值。

4）超链接标记

一个超链接唯一地指向另一个 Web 页，它由两部分组成：一部分是显示在本页面中的可被触发的超链接文本或图像（称为“热点”），另一部分是用来描述当超链接被触发后要链接到的 URL 信息。超链接标记的格式如下：

```
<a href="URL 信息">超链接文本或图像</a>
```

href 属性指出超链接的目标 URL 信息，分为以下三种情况。

① 若目标页面位于另外的主机或采用非 HTTP 协议，此时采用绝对 URL 格式，即协议名://主机名【/目录信息】。例如：

```
http://www.cernet.edu.cn
http://linux.cgi.com.cn/person/szj98/index.htm
ftp://ftp.njnet.edu.cn
mailto:wang@163.com
```

② 若目标页面位于本主机，可采用相对 URL 代替绝对 URL。例如，目标页面的 HTML 文件与本 HTML 文件位于同一子目录，名为 des1.htm，则超链接标记可简化为：

```
<a href="des1.htm">超链接文本</a>
```

又如：

```
<a href="../des2.htm">超链接文本</a>
```

③ 超链接的目标也可以是某个文件的特定位置（称为“锚点”，anchor）。此时，需要用超链接标记的 NAME 属性来定义超链接的引用名，格式为：

```
<a name="锚点名">文本或图像等页面元素</a>
```

注意，这里的文本或图像等页面元素并不被特殊显示，也不会触发超链接的跳转，它仅定义了一个超链接目标的引用名。当需要跳转到该目标时，将“#锚名”附加到 URL 之后即可。

超链接标记除了有必备的 href 属性外，还有一个很有用的属性 target，它指明目标页面显示的窗口。其含义如下。

① target=_blank　　//目标页面显示于一个新的浏览器窗口。

② target=_top　　//通常在框架中的超链接才设置该值，表示目标页面显示于整个浏览器窗口中，而不是显示于框架所在窗口中。

③ target=框架名　　//目标页面显示于指定框架所在的窗口。target 的默认值是本页面所在的浏览器窗口。关于框架见 3.2.5 节。

【例 3-3】三种 URL 应用示例。

```
<html><head><title>超链接 URL</title></head>
<body>
单击<a href="xp.htm" target=_blank><b>这里</b></a>可以见我的照片<br>
```

```
单击<a href="http://www.163.com"><b>这里</b></a>可以进入网易<br>
单击<a href="mailto:test@163.com"><b>这里</b></a>可以给我发信<br>
单击<a href="example3.htm#aaa"><b>这里</b></a>可以转到我的简历<br>
<a name="aaa">我的简历：</a></body></html>
```

5）列表标记和预定格式标记

列表标记用于产生不同格式的列表。有如下三种类型的列表：

① 无序列表（unordered list）：
列表项

② 有序列表（ordered list）：
列表项

③ 定义列表（definition list）：<dl>列表项</dl>

预定格式（preformatted）标记可以使信息按照 HTML 文件中编排的格式原样显示于浏览器中，该标记的格式为：

```
<pre>预定格式的信息</pre>
```

【例 3-4】三种列表标记应用示例。

```
<html><head><title>课表</title></head>
<body><b>今天我要上以下的课</b>
<ul><!--无序列表-->
  <li>局域网工程</li>
  <li>操作系统</li>
  <li>数据结构</li>
</ul>
<b>今天我要上以下的课</b>
<ol><!--有序列表-->
  <li>局域网工程</li>
  <li>操作系统</li>
  <li>数据结构</li>
</ol>
<dl><!--定义列表-->
<dt><b>局域网</b><!--定义标题--></dt>
<dd>局域网是指将小范围内的数据设备经过通信系统连接起来的计算机网络</dd>
</dl>
</body></html>
```

该 HTML 文件在浏览器中的显示效果如图 3-3 所示。

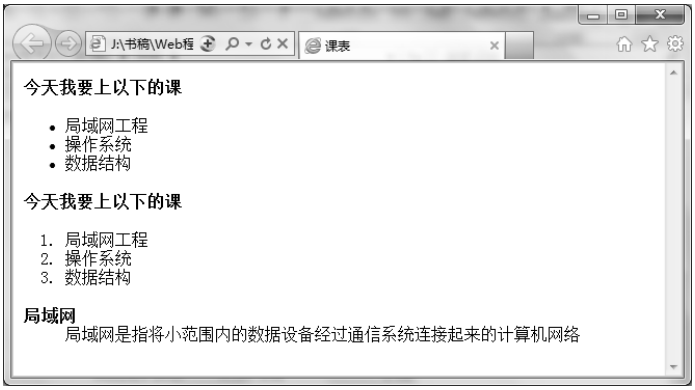


图 3-3　三种列表标记应用示例

3.2.3　表格（Table）

表格是最常用的页面元素，在页面中用表格来表示数据既直观又清晰，而且 HTML 表格的使

用非常灵活，许多较复杂的页面布局也可利用表格来完成。在 Internet 上浏览的许多页面都大量使用了表格。在 HTML 中，表格是由一个表格名称（标题）再加上一行或多行表格内容所构成的块状结构。

1．表格定义

表格定义的语法结构为：

```
<table>
  [<caption>标题内容</caption>]
      <tr>
          <td>表格内容</td>
          {<td>表格内容</td>}
      </tr>
      ...
</table>
```

<table>和</table>标记对界定表格结构的起始和结束；<caption>、</caption>标志是可选项，该标记中的内容是表格的标题；<tr>、</tr>界定一个表格行的开始和结束；一个表格行可以包含多个表格项，每个表格项的内容和显示特性由标记对<td>、</td>来定义。

2．表格属性

标记<table>、<caption>、<thead>、<tr>、<th>和<td>的属性用来定义表格的显示特性。其中<table>的属性描述整个表格的显示特性；<caption>标记描述表名；<thead>标记描述表的表头信息；行控制标记<tr>的属性定义该行的显示特性；标题栏标记<th>描述表格每列的标题信息；表格项控制标记<td>的属性定义该项的显示特性。利用它们丰富的属性可以设计出各种复杂的表格。标记<table>、<tr>、<td>的属性分别列于表 3-3、表 3-4 和表 3-5 中。

表 3-3　<table>标记属性表

属 性 名	取　值	含　义	默 认 值
border	整数	表格边框粗细，值为 0，表格没有边框；值越大，表格边框越粗	0
width	百分比	表格宽度，以相对于充满窗口的百分比计（如 60%）	100%
	整数	表格宽度，以屏幕像素点计	
cellpadding	整数	每个表项内容与表格边框之间的距离，以像素点为单位	0
cellspacing	整数	表格边框之间的距离，以像素点为单位	2
bordercolor	颜色值	表格边框的颜色	#000000
background	图像文件名	表格的背景图	无
align	left | center | right	表格的位置	left

表 3-4　<tr>标记属性表

属 性 名	取　值	含　义	默 认 值
align	left | center | right	本行各表格项的横向排列方式	left（左对齐）
bgcolor	颜色值	本行各表格项的背景色	#000000
valign	top | middle | bottom	本行各表格项的纵向排列方式	middle
width	百分比值 | 整数	本行宽度（受 table 的 width 属性值制约）	
height	整数	本行高度，以像素点为单位	

表 3-5　<td>标记属性表

属　性　名	取　　值	含　　义	默　认　值
align	left | center | right	本表格项的横向排列方式	left（左对齐）
bgcolor	颜色值	本表格项的背景色	#000000
valign	top | middle | bottom	本表格项的纵向排列方式	middle
width	百分比值 | 整数	本表格项宽度（受 table 和 tr 的 width 属性值制约）	
height	整数	本表格项高度，以像素点为单位（受 tr 的 height 属性值制约）	
background	图像文件名	本表格项的背景图像	无
colspan	整数	按列横向结合，如该值为 2，表示本表格项在宽度上占用两列	1
rowspan	整数	按行纵向结合，如该值为 2，表示本表格项在高度上占用两行	1

【例 3-5】一个简单的表格示例。本例只给出主要表格的部分文本，其余部分读者可以很容易补全。在浏览器中显示的结果如图 3-4 所示。

```
<table border=1 cellspacing=2 cellpadding=4>
<caption >物资列表</caption >
<thead> <tr><td>商品类别</td><td>数量</td></tr> </thead>
<tr><td>日用百货</td><td>10</td></tr>
<tr><td>电器</td><td>20</td></tr>
<tr><td>轿车</td><td>5</td></tr>
</table>
```

图 3-4　一个简单的表格示例

【例 3-6】一个较复杂的表格示例。其中每行列数及每列行数都不同，利用 td 标记的 colspan 和 rowspan 属性可对表格的单元格进行灵活的控制。在浏览器中显示的结果如图 3-5 所示。

图 3-5　一个复杂的表格示例

```
<html><head><title>复杂表格</title></head>
<body topmargin=4>
<table border=3 bordercolor=blue background="images/bock057.gif" align=center
cellspacing=3 cellpadding=6>
<caption>专业设置及在校生人数表</caption>
<tr align=center bgcolor=mediumturquoise>
<td><strong>学院名</strong></td>
<td colspan=4><strong>专业及人数</strong></td></tr>
<tr align=center><td rowspan=6>计算机学院</td>
<td colspan=4 bgcolor=ddeeff>计算机科学与技术专业</td></tr>
<tr align=center><td>2006 级</td><td>2007 级</td><td>2008 级</td><td>2009 级</td></tr>
```

```
<tr align=center ><td>300 人</td><td>200 人</td>
<td>150 人</td><td>120 人</td></tr>
<tr align=center ><td colspan=4 bgcolor=ddeeff >软件工程专业</td></tr>
<tr align=center><td>2006 级</td><td>2007 级</td><td>2008 级</td><td>2009 级</td></tr>
<tr align=center><td >100 人</td><td>80 人</td><td>50 人</td><td>40 人</td></tr>
<tr align=center><td rowspan=3>外语学院</td>
<td colspan=4 bgcolor=ddeeff >英语专业</td></tr>
<tr align=center><td>2006 级</td><td>2007 级</td><td>2008 级</td><td>2009 级</td></tr>
<tr align=center ><td >100 人</td><td>80 人</td><td>50 人</td><td>40 人</td></tr>
</table></body><html>
```

3.2.4 表单（Form）

表单提供图形用户界面的基本元素，包括按钮、文本框、单选钮、复选框等，是 HTML 实现交互功能的主要接口，用户通过表单向服务器提交数据。表单的使用包括两部分：一部分是用户界面，提供用户输入数据的元件；另一部分是处理程序，可以是客户端程序，在浏览器中执行，也可以是服务器处理程序，处理用户提交的数据，返回结果。本节仅介绍前一部分，即如何利用 HTML 提供的表单及相关标记生成用户界面，后一部分涉及 JavaScript、ASP.NET 程序设计，将在后续章节介绍。

1．表单定义

表单定义的语法如下：

```
<form method="get|post" action="处理程序名">
    [<input type=输入域种类  name=输入域名>]
    [teaxtarea 定义]
    [select 定义]
</form>
```

form 标记的属性含义如下。

（1）method：取值为 post 或 get。二者的区别是：get 方法将在浏览器的 URL 栏中显示所传递变量的值，而 post 方法则不显示；在服务器端的数据提取方式也不同。

（2）action：指出用户所提交的数据将由哪个服务器的哪个程序处理。可处理用户提交的数据的服务器程序种类较多，如 ASP 脚本程序、ASPX 程序、PHP 程序等。

form 的输入域有三类定义方式：input、textarea 和 select，定义方法和含义见下面的说明。

图 3-6　表单的输入域示例

2．表单的输入域

不同类型的输入域为用户提供灵活多样的输入数据的方式，表单的输入域有如下三类。

（1）以标记<input>定义的多种输入域，包括：text、radio、checkbox、password、hidden、button、submit、reset 和 file 等。

（2）以标记<textarea>定义的文本域。

（3）以标记<select>和<option>定义的下拉列表框。

【例 3-7】表单输入域的定义方法及使用示例。该 HTML 文件在浏览器中显示的效果如图 3-6 所示。

```
<html><head><title>表单使用</title></head>
```

```
<body><b>请选择您学习的方式</b><br>
<form method=get action="http://test.com/cgi-bin/run1">
<input type=radio checked>全日制在读
<input type=radio>走读
<input type=radio>函授<br><br>
<b>请选择您所要学习的课程</b><br>
<input type=checkbox value="yes" name="局域网工程" checked>局域网工程<br>
<input type=checkbox value="yes" name="操作系统">操作系统<br>
<input type=checkbox value="yes" name="数据结构">数据结构<br><br>
<b>请输入您的要求</b><br>
<textarea name="comment" rows=4 cols=50></textarea><br>
<input type=submit name="ok" value="提交">
<input type=reset name="re-input" value="重选"></form></body></html>
```

不同的输入域适用于接收用户不同的输入，常用的表单输入域列于表 3-6 中。

表 3-6　常用的表单输入域

输入域名称	说　明
text（文本框）	可输入一行文字。举例： <input type=text name="xm" size=10 value="">
radio（单选钮）	当有多个选项时，只能选其中一项。举例： 走<input type=radio name= "Rad" value="v1" checked> 留<input type=radio name="Rad" value="v2">
checkbox（复选框）	当有多个选项时，可以选其中多项。举例： 签字笔 <input type=checkbox name="ch1" checked> 钢笔<input type=checkbox name="ch2"> 圆珠笔<input type=checkbox name="ch3">
submit（提交按钮）	将数据传递给服务器。举例： <input type=submit name="ok" value="提交">
password（密码输入框）	用户输入的字符以“ * ”显示。举例： 输入密码：<input type=password size=12>
reset（重置按钮）	将用户输入的数据清除。举例： <input type=reset name="re-input" value="重选">
hidden（隐藏域）	在浏览器中不显示，但可通过程序取值或改变其值。主要用于浏览器向服务器传递数据而不想让浏览器用户知道的情形。举例： <input type=hidden name=hiddata value="HidValue" >
button（按钮）	普通按钮，按下后的操作需要由程序完成。举例： <input type=button value="去我的主页">
textarea（文本域）	可输入多行文字。举例： 请输入您的要求 <textarea name="comment" rows=4 cols=20></textarea>
select（下拉列表）	在多个可选项中选择，定义方法见下面的说明
file（文件域）	一般用于选择文件。举例： <input type="file" name="F1" size=20>

当提供给用户的选择项目较多时，为节省显示空间，可使用表单的下拉列表输入域。

定义下拉列表框使用<select>和<option>两个标记，其语法如下：

```
<select name=下拉列表框名  multiple>
    <option value=设定值>表项内容</option>
  ...
</select>
```

属性 multiple 是可选项，若定义该属性，则下拉列表中的多项都可被选中。

例如，下面的代码定义一个含有三个选项的下拉列表：

```
<form method=post action="http://test.com/cgi-bin/choice">
<select name="水果">
<option value="苹果">苹果</option>
<option value="梨子" selected>梨子</option>
<option value="香蕉">香蕉</option>
</select>
</form>
```

当用户需要上传文件时，可使用 file 输入域。文件域由一个文本框和一个“浏览”按钮组成，用户既可以在文本框中输入文件的路径和文件名，也可以通过单击“浏览”按钮从磁盘中查找和选择所需文件。创建文件域方法如下：

```
<input type="file"  属性="值"...>
```

属性主要包括 name、size 等，name 指出文件域名称，size 指出文件名输入框的宽度。

图 3-7　含有文件域的表单

【例 3-8】创建一个如图 3-7 所示的表单，其中包含文件域、“提交”按钮和“重置”按钮。

```
<html><head><title>文件域示例</title></head>
<body><form>
<table align=center bgcolor=#d6d3ce width=368>
<tr><th colspan=2 bgcolor=#0034FF><font color=#FFFFFF>文件域</font></th></tr>
<tr><td height=52 align=right>请选择文件：</td>
<td height=52><input type="file" name="F1" size=20></td></tr>
<tr align=center>
<td height=52 align=right><input type=submit value="提交" name="btnsubmit"></td>
<td height=52><input type=reset value="重置" name="btnreset"></td></tr>
</table></form></body></html>
```

3.2.5　框架（Frame）

框架又常称为帧。利用框架可以将浏览器显示窗口分割成多个相互独立的区域，每个区域可以显示独立的 HTML 页面。

1．一个简例

【例 3-9】一个应用框架的示例。其中包含三个 HTML 文件：main.htm 称为主文件，是包含<frame>标记的文件，它定义浏览器窗口被分割的方式，本例将窗口分为左、右两个子窗口，分别占窗口宽度的 15%和 85%；文件 frame1.htm、frame2.htm 分别是浏览器被分割的两个区域显示的页面文件。该 HTML 文件在浏览器中的显示效果如图 3-8 所示。

图 3-8　一个应用框架的示例

文件 main.htm 的内容：

```
<html>
<head><title>框架简例</title></head>
<frameset cols="15%,85%">
        <frame src="frame1.htm">
        <frame src="frame2.htm" scolling=no>
</frameset>
<noframes>
Please use a Web browser such as IE3.0 or Netscape Navigator
    to view this page in frames!
</noframes>
</html>
```

文件 frame1.htm 的内容：

```
<html>
<head><title>The document for the left frame</title></head>
<body bgcolor="aqua" text="#ff0000">左边子窗口的内容！</body>
</html>
```

文件 frame2.htm 的内容：

```
<html>
<head>
    <title>The document for the right frame</title>
</head>
<body>
    右边子窗口的内容！
</body>
</html>
```

2．框架定义

框架的定义较为特殊，首先需要确定如何分割窗口，然后建立描述窗口分割的主文件，再为每个框架建立相应的 HTML 文件。

主文件的定义方法是：

```
<html>
    <head>[头部标记]</head>
    <frameset>{<frameset>...</frameset>}
      <frame>
      <frame>
      ...
    </frameset>
   [<noframes>字符串</noframes>]
</html>
```

其中，标记<frameset>定义窗口分割的方式（横向或纵向）和大小，<frameset>可以嵌套，内层的<frameset>表示对已分割的窗口再进行分割的方式和大小。<frame>标记指明框架所对应的 HTML 文件。<frame>标记的个数应与其所属的<frameset>标记分割的框架数目相同，与窗口的对应关系是按排列顺序逐个对应。<noframes>标记定义了若浏览器不支持框架时所显示的内容。目前浏览器都已支持框架。<framset>和<frame>标记的主要属性列于表 3-7 和表 3-8 中，它们描述分割窗口的特性及框架中页面显示的特性。

表 3-7　<framset>标记属性表

属　性　名	取　　值	含　　义	默　认　值
rows	百分比	将窗口上、下（横向）分割，给出每个框架高度占整个窗口高度的百分比。例如：“ 25%,75% ”表示将窗口分为上、下两个框架，高度分别为总窗口高度的 25%和 75%。值的一部分也可用“ * ”表示，例如“ 25%,* ”，表示最后一个框架的高度是除去其他框架已用的高度	无
	整数	将窗口上、下（横向）分割，给出每个框架高度的像素点数。例如：“ 100,600 ”表示将窗口分为上、下两个框架，高度分别为 100 和 600 个像素点。值的一部分也可用“ * ”表示，含义同上	
cols	百分比	将窗口左、右（纵向）分割，值的格式和含义与“ rows ”属性类似	无
	整数		
frameborder	yes|no	框架边框是否显示	yes
bordercolor	颜色值	框架边框颜色	gray（灰）

表 3-8　<frame>标记属性表

属　性　名	取　　值	含　　义	默　认　值
src	HTML 文件名	框架对应的 HTML 文件	无
name	字符串	框架的名字，可在程序和<a>标记的 target 属性中引用	无
noresize	无	不允许用户改变框架窗口大小	无
scrolling	yes|no|auto	框架边框是否出现滚动条	auto
marginwidth	整数	框架左、右边缘像素点数	0
marginheight	整数	框架上、下边缘像素点数	0

【例 3-10】利用框架将窗口分成三个子窗口，分别命名为 win001、win002 和 win003，子窗口 win001 对应的 HTML 中设置了两个超链接，用户单击这两个超链接后目标 URL 将在子窗口 win002 中显示。

主文件：

```
<html><head><title>较复杂的框架例子</title></head>
    <frameset rows="360,*" bordercolor="green">
        <frameset cols="30%,*">
            <frame src="frame1.htm" scrolling="no" name="win001">
            <frame src="frame2.htm" name="win002">
        </frameset>
    <frame src="frame3.htm" noresize marginwidth=5 name="win003">
    </frameset>
    <noframes>
      Please use a Web browser such as IE3.0 or Netscape Navigator
      to view this page in frames!
    </noframes>
</html>
```

文件 frame1.htm：

```
<html><head><title>左边框架</title></head>
<body><a href="frame2.htm" target="win002">第一章</a><br><br>
<a href="第二章.htm" target="win002">第二章</a></body></html>
```

文件 frame2.htm：

```
<html><head><title>第一章</title></head>
<body><h1>第一章　绪论</h1><br>本章简述课程的要点...<br><br>
<a href="frame2.htm">返回</a></body></html>
```

文件 frame3.htm：

```
<html><head><title>第三个框架</title></head>
<body><h2>联系人地址：
test@gu.com</h2></body></html>
```

图 3-9　将超链接的目标显示于另一框架中

例 3-10 在浏览器中显示的效果如图 3-9 所示。

例 3-10 在上左子窗口对应的文件 frame1.htm 中设置了两个超链接，它们被触发后，相应的目标页面将显示于上右子窗口（名为“ win002 ”）中，这是通过在文件 frame1.htm 的标记<a>中设置 target 属性来指定的。这种方法在页面设计中被广泛使用，它可以保持超链接不被目标文件覆盖。

3.3　可扩展标记语言 XML

可扩展标记语言 XML（eXtensible Markup Language）是为了克服 HTML 缺乏灵活性和伸缩性的缺点及 SGML 过于复杂、不利于软件应用的缺点而发展起来的一种元标记语言。SGML 功能强大，但是为能实现强大的功能，要做非常复杂的准备工作。首先要创建一个文档类型定义，在该定义中给出标记语言的定义和全部规则，然后再编写 SGML 文档，并把文档类型定义和 SGML 文档一起发送，才能保证用户定义的标记能够被理解。HTML 简单易学，但也有不足之处：首先，HTML 的标记是固定的，不允许用户创建自己的标记；其次，HTML 中标记的作用是描述数据的显示方式，并且只能由浏览器进行处理；另外，在 HTML 中，所有标记都独立存在，无法显示数据之间的层次关系。

XML 吸取了两者的优点，摒弃了它们的缺点，已成为互联网标准的重要组成部分。在 XML 中，用户可以根据所要描述的数据元素定义不同的标签，表达各种丰富的内容和意义。

3.3.1　XML 概述

XML 是一种数据存储语言，它使用一系列简单标记描述数据。XML 同时也是一组规范，用户都遵守这组规范进行开发，这样，不同计算机系统之间就可以相互交流信息。

XML 继承了 SGML 和 HTML 的功能，是一种用于定义标记的语言，又称为“元语言”。创建一个 XML 文档时，用户需要根据描述的数据自己定义各种标记。

1．XML 与 HTML 的比较

先看一个示例。

【例 3-11】XML 与 HTML 的比较示例。

```
<BODY> Here we have some text
<H1> This is a heading </H1>
This bit is normal text
<B> This is some bold text </B>
And finally some more normal text </BODY>
```

如果上面的代码是 HTML 文档，将其加载到浏览器中，就会显示如图 3-10 所示的结果，其作用是格式化文档。但是，如果上面的代码是 XML 文档，那么其中的标记就不具有任何含义，其内容只是说明如下内容：

（1）有一个名为 BODY 的标记，在这个标记里面有一些文本；

（2）有一个名为 H1 的标记，在这个标记里有一些文本；

（3）有一个名为 B 的标记，在这个标记里有一些文本。

如果例 3-11 的代码作为一个 XML 文档（文件扩展名为.xml）加载到 IE 浏览器中，其结果如图 3-11 所示。浏览器只是把这些标记原封不动地显示出来。

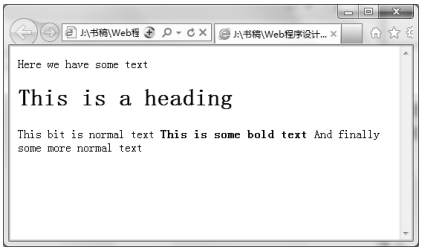


图 3-10　浏览器中显示的 HTML 文档

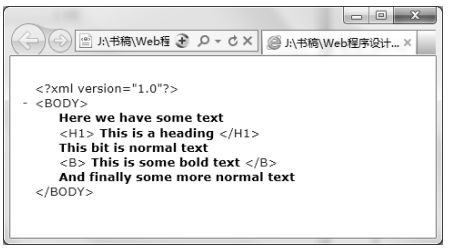


图 3-11　浏览器中显示的 XML 文档

HTML 提供了固定的预定义元素集，可以使用这些元素来标记一个 Web 页的各个组成部分。而 XML 没有预定义的元素，用户可以创建自己的元素，并自行命名。XML 标记是可以扩展的，用户可以根据需要定义新的标记。XML 标记用于描述文本的结构，而不是用于描述如何显示文本。表 3-9 给出了 HTML 与 XML 的主要不同点比较。

表 3-9　HTML 与 XML 的比较

比 较 内 容	HTML	XML
可扩展性	不具有扩展性	是元标记语言，可用于定义新的标记语言
侧重点	侧重于信息的表现	侧重于结构化地描述信息
语法要求	较宽松，不要求标记嵌套、配对等	语法严谨，严格要求标记嵌套、配对、遵循 XML 数据结构（DTD 树形结构、XML Schema）
可读性与可维护性	难于阅读与维护	结构清晰，便于阅读与维护
数据与显示关系	内容描述与显示方式一体化	内容描述与显示方式分离
大小写敏感	不区分大小写	区分大小写

目前，XML 并没有取代 HTML，还在与 HTML 一起使用。XML 极大地扩展了 Web 页的能力，使 Web 页能传递任意类型的文档、对数据进行各类管理，以及操作高度结构化的信息，并且 XML 可以与 HTML 进行互操作。

2．XML 的特性

（1）实现应用程序之间的数据交换。XML 是跨平台的，它提供在不同应用程序之间进行数据交换的公共标准，是一种公共的交互平台。

（2）数据与显示分离。一个 XML 文件并不能决定数据的显示样式，数据的显示部分需要由其他语言来确定（如由 CSS 样式来确定），这样就可按用户意愿给一份数据设置多种样式，如图 3-12 所示。

（3）数据分布式处理。XML 文档通过网络传送给用户后，用户可通过各类应用软件从 XML 文档中提取数据，进而对数据进行各种处理，如编辑、排序等。XML 文档对象模型（Document Object Model，DOM）允许使用脚本语言或其他编程语言处理 XML 数据，从而使得数据可以在各用户端处理，而不必都集中在 Web 服务器中，实现了数据的分布式处理，如图 3-13 所示。

此外 XML 还具有可扩展性强、易学易用等特点。

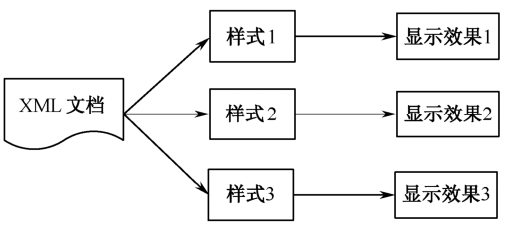


图 3-12　XML 显示样式示意图

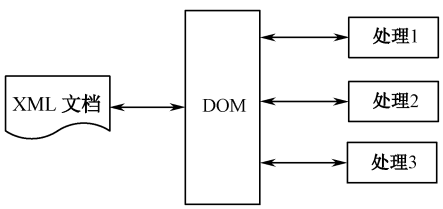


图 3-13　XML 分布式数据处理示意图

3．XML 文档处理流程

一般地，一个 XML 文档的处理流程如图 3-14 所示。

图 3-14　XML 文档处理流程

整个处理过程分为如下三个阶段。

（1）编辑。使用通用的文字处理软件或专用的 XML 编辑工具生成 XML 文档。

（2）解析。对 XML 文档进行语法分析、合法性检查。读取其中的内容，通常以树形结构交给后续的应用程序进行处理，后续程序通常为浏览器或其他应用程序。

（3）浏览。将由 XML 解析器传来的 XML 树形结构以用户需要的格式显示或处理。

4．XML 工具

针对图 3-14 所示的 XML 文档处理流程，XML 的开发应用环境包括 XML 编辑工具、解析工具和浏览工具等。

（1）XML 编辑工具。XML 文档的编辑与保存都是纯文本格式，因此使用通用的文本编辑软件，如 Windows 记事本、写字板、MS Office 等，都可以创建 XML 文档。但是这些通用文字处理软件不能真正理解 XML。专用的 XML 编辑器可以理解 XML，将它们显示为树形结构。常见的专用 XML 编辑器有 XMLwriter、XML Spy、XML Pro、Visual XML 等。

（2）XML 解析工具。也称解析器（Parser），它是 XML 的语法分析程序。其主要功能是读取 XML 文档并检查其文档结构是否完整，是否有结构上的错误；对于结构正确的文档，读出其内容，交给后续程序去处理。常见的 XML 解析器有 Apache Xeces、MSXML 等。

（3）XML 浏览工具。XML 解析器会将 XML 文档结构和内容传输给用户端应用程序。在大多数情况下，用户端应用程序可能是浏览器或其他应用程序（如将数据转换后存入数据库）。如果是浏览器，数据就会显示给用户。

3.3.2　XML 文档的编写

1．XML 文档的组成

XML 定义了如何标记文档的一套规则。可根据需要给标记取任何名字，如<BOOK>、<TITLE>、<AUTHOR>等。下面是一个格式正确的 XML 文档示例。

【例 3-12】一个 XML 文档示例。

```
<?xml version='1.0' standalone='yes' ?>
<?xml-stylesheet type="text/css" href="Example.css"?>
<INVENTORY>
        <BOOK>
            <TITLE>The Adventures of Huckleberry Finn</TITLE>
            <AUTHOR>Mark Twain</AUTHOR>
            <BINDING>mass market paperback</BINDING>
            <PAGES>298</PAGES>
            <PRICE>$5.49</PRICE>
        </BOOK>
        <BOOK>
            <TITLE>Leaves of Grass</TITLE>
            <AUTHOR>Walt Whitman</AUTHOR>
            <BINDING>hardcover</BINDING>
            <PAGES>462</PAGES>
            <PRICE>$7.75</PRICE>
        </BOOK>
        <BOOK>
            <TITLE>The Legend of Sleepy Hollow</TITLE>
            <AUTHOR>Washington Irving</AUTHOR>
```

```
        <BINDING>mass market paperback</BINDING>
        <PAGES>98</PAGES>
        <PRICE>$2.95</PRICE>
    </BOOK>
    <BOOK>
        <TITLE>The Marble Faun</TITLE>
        <AUTHOR>Nathaniel Hawthorne</AUTHOR>
        <BINDING>trade paperback</BINDING>
        <PAGES>473</PAGES>
        <PRICE>$10.95</PRICE>
    </BOOK>
    <BOOK>
        <TITLE>Moby-Dick</TITLE>
        <AUTHOR>Herman Melville</AUTHOR>
        <BINDING>hardcover</BINDING>
        <PAGES>724</PAGES>
        <PRICE>$9.95</PRICE>
    </BOOK>
    <BOOK>
        <TITLE>The Portrait of a Lady</TITLE>
        <AUTHOR>Henry James</AUTHOR>
        <BINDING>mass market paperback</BINDING>
        <PAGES>256</PAGES>
        <PRICE>$4.95</PRICE>
    </BOOK>
    <BOOK>
        <TITLE>The Scarlet Letter</TITLE>
        <AUTHOR>Nathaniel Hawthorne</AUTHOR>
        <BINDING>trade paperback</BINDING>
        <PAGES>253</PAGES>
        <PRICE>$4.25</PRICE>
    </BOOK>
    <BOOK>
        <TITLE>The Turn of the Screw</TITLE>
        <AUTHOR>Henry James</AUTHOR>
        <BINDING>trade paperback</BINDING>
        <PAGES>384</PAGES>
        <PRICE>$3.35</PRICE>
    </BOOK>
</INVENTORY>
```

可见，XML 文档中不包含格式信息，而是定义了<BOOK>、<TITLE>、<AUTHOR>等标记来表示数据的真实含义。XML 标记就是定界符（即“ < > ”）及用定界符括起来的文本。

与 HTML 类似，在 XML 中，标记也是成对出现的。处于前面的是起始标记，如<BOOK>、<TITLE>、<AUTHOR>等，而位于后面的是结束标记，如</BOOK>、</TITLE>、</AUTHOR>等。与 HTML 不同的是，在 XML 中，结束标记是不可省略的。另外，标记是区分大小写的，如<BOOK>和<Book>是两个不同的标记。标记和起始/结束标记之间的文字结合在一起构成元素。所有元素都可以有自己的属性，属性采用“ 属性/值 ”对的方式写在标记中。

一个 XML 文档主要由两部分组成：序言和文档元素。在文档元素之后可以包括注释、处理指令和空格等。

（1）序言

例 3-12 给出的示例文档的序言由两行组成：第一行是 XML 声明，它说明这是一个 XML 文档，并且给出了版本号。XML 声明还包括一个独立文档声明（standalone = 'yes'）。这个声明可以被某些 XML 文档用来简化文档处理。XML 声明是可选的。第二行包括一条处理指令，该处理指令告诉应用程序使用文件 Example.css 中的 CSS，处理指令的目的是给有关 XML 应用程序提供信息。

（2）文档元素

XML 文档元素是以树形分层结构排列的，元素可以嵌套在其他元素中。文档必须只有一个顶层元素，称为文档元素（也称根元素），类似于 HTML 页中的 BODY 元素，其他所有元素都嵌套在其中。在例 3-12 中，文档元素是 INVENTORY，其起始标记是<INVENTORY>，结束标记是</INVENTORY>，内容是 8 个嵌套的 BOOK 元素。

在 XML 文档中，元素指出了文档的逻辑结构，并且包含了文档的信息内容。一个典型的元素有起始标记、元素内容和结束标记。元素内容可以是字符、数据、其他（嵌套的）元素或两者的组合。

2．创建 XML 文档的基本规则

一个格式正确的文档是符合最小规则集的文档，它可以被浏览器或其他程序处理。下面是创建格式正确的 XML 文档的一些基本规则。

（1）文档必须有一个顶层元素（文档元素或根元素），所有其他元素必须嵌入其中。

（2）元素必须被正确地嵌套。也就是说，如果一个元素在另一个元素中开始，那么它必须在同一个元素中结束。

（3）每一个元素必须同时拥有起始标记和结束标记。与 HTML 不同，XML 不允许忽略结束标记，即使浏览器能够推测出元素在何处结束时也是如此。

（4）起始标记中的元素类型名必须与相应结束标记中的名称完全匹配。

（5）元素类型名是区分大小写的。实际上，XML 标记中的所有文本都是区分大小写的。例如，下列元素是非法的，因为起始标记的类型名与结束标记的类型名不匹配。

```
<TITLE>Leaves of Grass</Title>
```

3．元素内容的类型

元素内容是起始标记和结束标记之间的文本。其中可以包括嵌套元素和字符数据两种类型。当给元素添加字符数据时，用户无法插入“ < ”符号、“ & ”符号或字符串“]]> ”作为字符数据的一部分，因为 XML 解析器会把“ < ”解释为嵌套元素的开始，把“ & ”解释为一个实体引用或字符引用的开始，把“]]> ”解释为 CDATA 节的结束。如果要想把“ < ”和“ & ”作为字符数据的一部分，可以使用 CDATA 节。还可以通过字符引用插入任意字符，或通过使用预定义的通用实体引用来插入某个字符（如“ < ”或“ & ”）。有关实体引用、字符引用和 CDATA 节的内容将在后面介绍。

4．给元素添加属性

在一个元素的起始标记中，可以包含一个或多个属性。属性由属性名、等号及属性值组成。属性名可以由用户任意定义。例如，下面的 PRICE 元素包含一个名为 Type 的属性，它被赋值为 retail。

```
<PRICE    Type= "retail" > $12.50 </PRICE>
```

给元素添加属性是为元素提供信息的一种方法。当使用 CSS 显示 XML 文档时，浏览器不会显示属性及它们的值。但是，若使用数据绑定、HTML 页中的脚本或者 XSL 样式表显示 XML 文

档时，则可以访问属性及其值。

5．处理指令的使用

处理指令的一般形式为：

```
< ? target instruction ? >
```

这里，target 是指令所指向的应用名称。名称必须以字母或下画线开头，后面跟若干个数字、字母、点、连字符或下画线。“ xml ”是保留名称，它是处理指令的一种类型。例如：

```
<?xml version='1.0' standalone='yes' ?>
```

在 XML 文档中使用的处理指令取决于读取文档的处理器。如果使用 IE5.0 作为 XML 处理器，那么处理指令主要有如下两种用途。

（1）可以使用标准的、预留处理指令来告诉 IE5.0 怎样处理和显示文档。

（2）如果编写了 Web 页脚本用于处理和显示 XML 文档，那么可以在文档中插入任意非保留的处理指令。

6．CDATA 节的使用

CDATA 节以字符“ <![CDATA[”开始，并以字符“]]> ”结束。在这两个限定字符组之间，可以输入包括“ < ”或“ & ”的任意字符，“]]> ”除外。CDATA 节中的所有字符都会被当作元素中字符数据的常量部分，而不是 XML 标记。在任何出现字符数据的地方都可以插入 CDATA 节。下面是一个合法 CDATA 节的例子：

```
<?xml version= "1.0" ?>
<MUSICAL>
 <TITLE_PAGE>
   <! [CDATA[
       <oklahoma!>    By   Rogers & Hammerstein    ]]>
 </TITLE_PAGE>
</MUSICAL>
```

通过以上学习，我们了解了 XML 文件的基本规则，这样就能够编写一个规范的 XML 文件了。但是，这个 XML 文件可能并不能准确地描述客观事物，这是因为还没有一个更加详细的规范来约束 XML 文件，即还缺乏 XML 数据的底层数据结构。为此，人们制定了两个对 XML 文件的约束规范：文档类型定义 DTD（Document Type Definition）、XML Schema。其中，XML Schema 是继 DTD 之后的第二代用于描述 XML 文件的标准，功能更为强大。我们把符合 XML 语法规则的 XML 文件称为规范的 XML 文件，也称为良构的 XML 文件，而将符合 DTD 或 XML Schema 规范的 XML 文件称为有效的 XML 文件。限于篇幅，本书不再详细介绍 DTD 和 XML Schema，有兴趣的读者可参阅有关资料。

3.3.3 XML 文档的显示

如果需要将 XML 文档在浏览器中按特定的格式显示出来，必须要有另一个文件告诉浏览器如何显示。XML 文档由专门的样式文档来执行，可以是层叠样式表 CSS 或是可扩展样式表语言 XSL（eXtensible Stylesheet Language）。

1．使用 CSS 样式表显示 XML 文档

可以直接在浏览器中打开 XML 文档，就像打开一个 HTML Web 页一样。如果 XML 文档没有包含指向样式表的链接，那么浏览器只显示整个文档的文本，包括标记和字符数据。浏览器用带

颜色的代码来区分不同的文档组成部分，并且以收缩和扩展树的形式显示文档元素，以便清楚地指出文档的逻辑结构并允许详细地查看图层。

如果 XML 文档包含指向样式表的链接，那么浏览器只显示文档元素的字符数据，并根据样式表中指定的规则格式化数据。可以使用层叠样式表 CSS 或可扩展样式表语言 XSL 编写样式表。使用层叠样式表 CSS 显示 XML 文档有两个基本步骤：(1) 创建 CSS 样式表文件；(2) 链接 CSS 样式表到 XML 文档。其中，创建 CSS 样式表文件的问题将在 4.3 节中介绍。例如，对于例 3-12，可以建立如下的样式表文件 Example.css：

```
BOOK
   {display:block; margin-top:12pt; font-size:10pt}
TITLE
   {font-style:italic}
AUTHOR
   {font-weight:bold}
```

要链接层叠样式表到 XML 文档，则需要插入保留的 xml-stylesheet 处理指令到 XML 文档中。这个处理指令有如下所示的通用格式，其中 CSSFilePath 指示样式表文件位置的 URL：

```
<？xml-stylesheet type="text/css" href=CSSFilePath ?>
```

例如，例 3-12 中的 XML 文档中包含如下处理指令：

```
<？xml-stylesheet type="text/css" href="Example.css" ?>
```

例 3-12 的 XML 文档在浏览器中的显示效果如图 3-15 所示。

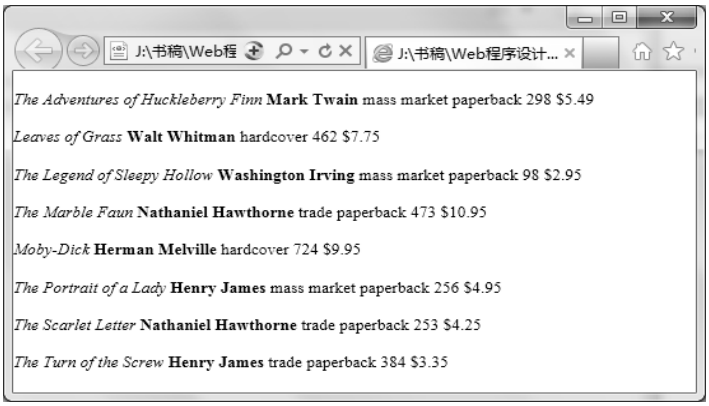


图 3-15　应用 CSS 样式文件显示 XML 文档

2．使用 XSL 样式文件显示 XML 文档

与 CSS 样式表类似，一个 XSL 样式表链接到一个 XML 文档后，浏览器就可以显示 XML 数据了。但对于显示 XML 来说，XSL 样式表的功能远比 CSS 强大和灵活。CSS 只允许指定每个 XML 元素的格式，而 XSL 样式表允许对输出进行完整的控制。特别地，XSL 允许精确地选择想要显示的 XML 数据，允许用任意顺序和排列方式表示数据，并且可以修改或添加信息。XSL 可以访问所有的 XML 组件（包括元素、属性、注释和处理指令），允许排序和过滤 XML 数据，允许在样式表中包含脚本，同时提供一组方法以便处理信息。

(1) 使用 XSL 样式表的基本步骤

使用 XSL 样式表显示 XML 文档有如下两个基本步骤。

① 创建 XSL 样式表文件。XSL 是 XML 的一个应用，即一个 XSL 样式表是一个遵守 XSL 规则的格式正确的 XML 文档，其扩展名为.xsl。

② 链接 XSL 样式表到 XML 文档。通过在 XML 文档中包含一个 xml-stylesheet 处理指令，链

接 XSL 样式表到 XML 文档。指令格式如下：

```
< ? xml-stylesheet type= "text/xsl" href=XSLFilePath? >
```

这里，XSLFilePath 是一个带引号的 URL，它指出样式表文件的位置。例如：

```
< ? xml-stylesheet type= "text/xsl" href= "Example.xsl" ? >
```

通常，把 xml-stylesheet 处理指令添加到 XML 文档的序言中，并放在 XML 声明之后。

如果一个 XSL 样式表已链接到 XML 文档，那么就可以直接在浏览器中打开该文档，而且浏览器将使用样式表中的转换指令显示 XML 文档。与层叠样式表不同，如果链接了多个 XSL 样式表到 XML 文档，那么浏览器将使用第一个而忽略其他的。如果同时链接了一个 CSS 和 XSL 样式表到 XML 文档，那么浏览器将只使用 XSL 样式表。

（2）使用 XSL 模板显示 XML 文档

XSL 样式表包含一个或多个模板（Template），每个模板包含显示 XML 文档中元素的某个分支的信息。每个 XSL 样式表必须有一个如下所示的文档元素：

```
< xsl:stylesheet xmlns:xsl= "http://www.w3.org/1999/XSL/Transform">
< ! – one or more template elements ... -- >
< /xsl:stylesheet >
```

XML 样式表的文档元素 xsl:stylesheet 必须包含一个或多个 XSL 模板元素, 也可把模板元素简称为模板。它具有下列形式：

```
< xsl:template match= "/" >
   < ! – child elements ... -- >
< / xsl:template >
```

浏览器使用模板来显示样式表所链接的 XML 文档中元素层次结构的某个分支。模板的 match 属性指定某个分支（match 属性类似于 CSS 规则的选择器）。match 属性的值被称为模式（pattern）。模式（“ / ”）代表整个 XML 文档的根。每一个 XSL 样式表必须包含一个 match 属性值被设置为“ / ”的模板，还可以包括一个或多个包含指令的附加模板来显示 XML 文档元素结构的特定子分支。每个模板必须有一个与特定分支匹配的模式。

【例 3-13】一个 XSL 样式表和 XML 文档示例。

本例的 XML 文档将例 3-12 中的前两行换为如下内容，其余不变。

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<?xml-stylesheet type="text/xsl" href="book1.xsl"?>
```

XSL 样式表 book1.xsl 清单如下：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
 <body>
     <h2>Book Inventory</h2>
     <table border="1">
         <tr bgcolor="#8acc33">
             <th>Title</th>
             <th>Author</th>
             <th>Binding type</th>
             <th>Pages</th>
             <th>Price</th>
         </tr>
         <tr>
             <td><xsl:value-of select="INVENTORY/BOOK/TITLE"/></td>
             <td><xsl:value-of select="INVENTORY/BOOK/AUTHOR"/></td>
```

```
                    <td><xsl:value-of select="INVENTORY/BOOK/BINDING"/></td>
                    <td><xsl:value-of select="INVENTORY/BOOK/PAGES"/></td>
                    <td><xsl:value-of select="INVENTORY/BOOK/PRICE"/></td>
                </tr>
            </table>
        </body>
    </html>
</xsl:template>
</xsl:stylesheet>
```

<xsl:value-of> 元素用于提取某个选定节点的值。例 3-13 的 XML 文档在浏览器中的显示效果如图 3-16 所示。

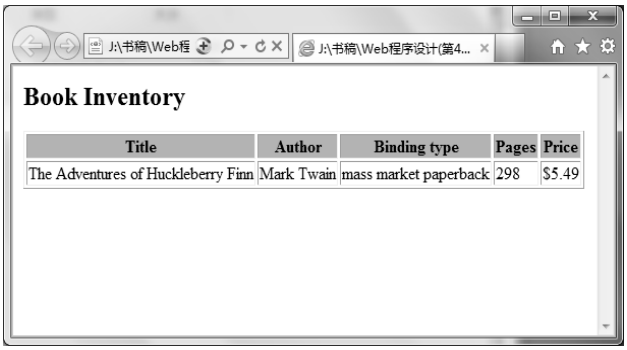


图 3-16　应用 XSL 样式文件显示 XML 文档

注意，赋予每个 select 属性的模式都是以文档元素开头的 XPath 表达式，即各层节点名以“ / ”连接形成的路径表达式，例如“ INVENTORY/BOOK/AUTHOR ”。每个模式需逐级匹配，如“ INVENTORY/BOOK/AUTHOR ”需依次匹配 INVENTORY、BOOK、AUTHOR。当 XML 文档中有多个元素时，浏览器只提取第一个匹配的元素。由图 3-16 可以看出，结果中只包含一个 BOOK 元素。

若要提取所有匹配的元素，需使用 XSL 的 for-each 元素（它会重复输出所包含的元素）。

例 3-14 的 XSL 样式表采用这种技术。

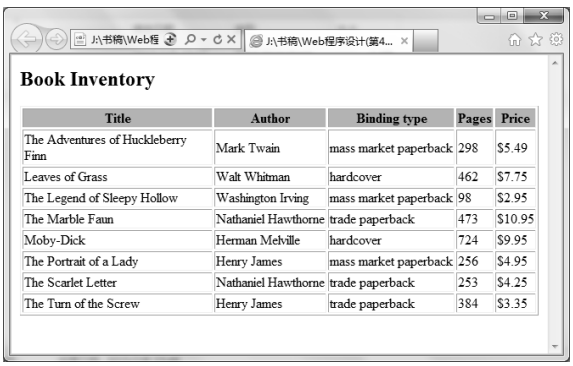
【例 3-14】使用 for-each 元素的 XSL 样式文件示例。

XSL 样式表 book2.xsl 清单如下：

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xsl:stylesheet version="1.0" xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
<html>
    <body>
        <h2>Book Inventory</h2>
            <table border="1">
                <tr bgcolor="#8acc33">
                    <th>Title</th>
                    <th>Author</th>
                    <th>Binding type</th>
                    <th>Pages</th>
                    <th>Price</th>
                </tr>
                 <xsl:for-each select="INVENTORY/BOOK">
                 <tr>
                    <td><xsl:value-of select="TITLE"/></td>
                    <td><xsl:value-of select="AUTHOR"/></td>
                    <td><xsl:value-of select="BINDING"/></td>
```

```
                    <td><xsl:value-of select="PAGES"/></td>
                    <td><xsl:value-of select="PRICE"/></td>
                  </tr>
                </xsl:for-each>
              </table>
            </body>
          </html>
        </xsl:template>
        </xsl:stylesheet>
```

<xsl:for-each> 元素可用于选取指定的节点集中的每个 XML 元素。例 3-14 的 XML 文档在浏览器中的显示效果如图 3-17 所示。



Book Inventory

Title	Author	Binding type	Pages	Price
The Adventures of Huckleberry Finn	Mark Twain	mass market paperback	298	$5.49
Leaves of Grass	Walt Whitman	hardcover	462	$7.75
The Legend of Sleepy Hollow	Washington Irving	mass market paperback	98	$2.95
The Marble Faun	Nathaniel Hawthorne	trade paperback	473	$10.95
Moby-Dick	Herman Melville	hardcover	724	$9.95
The Portrait of a Lady	Henry James	mass market paperback	256	$4.95
The Scarlet Letter	Nathaniel Hawthorne	trade paperback	253	$4.25
The Turn of the Screw	Henry James	trade paperback	384	$3.35

图 3-17　例 3-14 的 XML 页面

本 章 小 结

本章介绍了 HTML 和 XML 两种语言。

超文本标记语言 HTML 是在万维网上建立超文本文件的语言，它通过标记与属性对一段文本进行描述，并提供由一个文件到另一个文件，或在一个文件内部不同部分之间的链接。HTML 文件是普通的文本文件，与平台无关，可用任何文本编辑器进行编辑，文件的扩展名为.htm 或.html。HTML 标记描述了文档的结构，是区分文本各个部分的分界符，用于将 HTML 文档划分成不同的逻辑部分，它与属性一起向浏览器提供该文档的格式化信息，以向浏览器传递文档的外观特征。HTML 标记分为头部标记和体部标记两大类：头部标记主要用于说明有关 HTML 本身的信息及体部样式；体部标记众多，按其用途可分为基本格式控制标记、列表控制标记、表格控制标记、表单控制标记和框架控制标记。

XML 是可扩展标记语言，其用途主要有两个：一是作为元标记语言，定义各种实例标记语言标准；二是作为标准交换语言，起描述交换数据的作用。本章介绍了 XML 的基本技术，主要内容包括 XML 文档的编写方法和规则及在 Web 浏览器中显示 XML 文档的技术。一个 XML 文档主要由两部分组成：序言和文档元素。每一个 XML 文档必须有正确的格式。在浏览器中显示 XML 文档主要有两种技术：使用 CSS 样式表或 XSL 样式表显示 XML 文档。

习　题　3

3.1　简述超文本标记语言 HTML 的特点。

3.2　试述 HTML 文件的结构。

3.3　简述 HTML 表格（Table）的创建要点。
3.4　什么是表单（Form）？在 HTML 中如何创建表单？
3.5　在 HTML 中使用框架（Frame）结构的目的是什么？
3.6　试用 HTML 语言设计一个简单的个人主页，内容包括简介、兴趣爱好、特长等。
3.7　XML 的特点是什么？
3.8　XML 被称为“元标记”语言，试解释其含义。
3.9　试比较 HTML 与 XML。
3.10　试用 XML 语言编写一个班级通讯录网页。
3.11　试说明在浏览器中显示 XML 文档的主要技术。

上机实验 3

实验 3.1　《Web 程序设计》课程网站主页面设计。
【目的】
（1）掌握 HTML 常用标记的用法。
（2）掌握应用表格进行页面布局的方法。
【内容】设计如图 3-18 所示的《Web 程序设计》课程网站主页面。

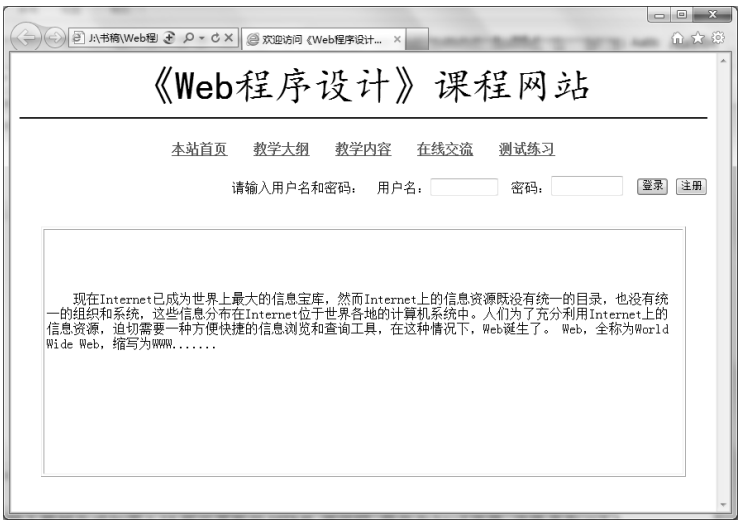


图 3-18　《Web 程序设计》课程网站主页面

【步骤】
（1）打开记事本程序。
（2）输入能够生成如图 3-18 所示页面的 HTML 源代码，保存为.html 文件，文件名为 ex3-1。
（3）双击 ex3-1.html 文件，在浏览器中查看结果。

实验 3.2　设计《Web 程序设计》课程网站“教学内容”功能。
【目的】
（1）进一步熟练使用 HTML 语言的常用标记。
（2）掌握应用框架进行页面布局的方法。
【内容】设计如图 3-19 所示的《Web 程序设计》课程网站的“教学内容”框架页面。

要求：在“教学内容”页面中，左边为各章标题，每章标题都是超链接，单击章标题后，将在右边显示该章的教学内容。

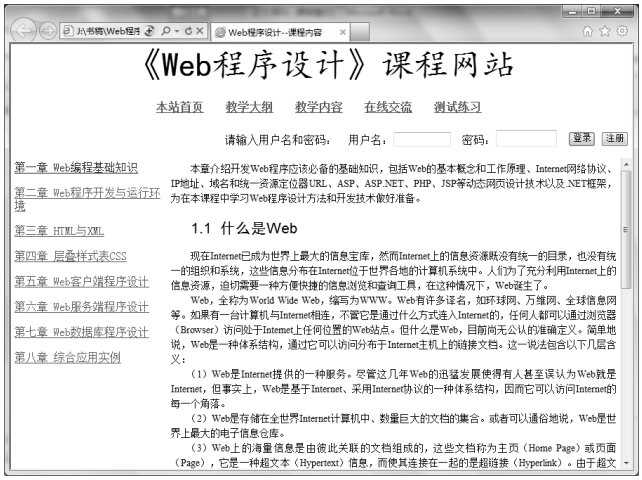


图 3-19 《Web 程序设计》课程网站“教学内容”页面

【步骤】

（1）打开记事本程序。

（2）输入能够生成如图 3-19 所示页面的 HTML 源代码，分别保存为 study.html、title.html、chapter.html、chap1.html～chap8.html 文件。

（3）双击 study.html 文件，在浏览器中查看结果。

实验 3.3 用 XML 语言设计网页。

【目的】掌握用 XML 语言设计网页的基本过程。

【内容】使用 XML 语言设计如图 3-20 所示的班级通讯录网页，并在浏览器中显示。

图 3-20 班级通讯录网页

【步骤】

（1）在记事本中输入能够生成如图 3-20 所示页面的 XSL 样式表文件，保存为 ex3-3.xsl 文件。

（2）在记事本中输入能够生成如图 3-20 所示页面的 XML 文件，保存为 ex3-3.xml 文件。

（3）双击 ex3-3.xml 文件，在浏览器中查看结果。

第 4 章　层叠样式表 CSS

HTML 中的显示特性是通过标记的属性来设置的，一旦设置就难以变化，且不能由程序控制，具有很大的局限性。CSS（Cascading Style Sheets，层叠样式表）是 W3C 协会为弥补 HTML 在显示属性设定上的不足而制定的一套扩展样式标准。它扩充了 HTML 标记的属性设定，称为 CSS 样式，并且通过脚本程序控制，可以使页面的表现方式更为灵活，更具动态特性。CSS 可提供多种样式，以减少 GIF 动画的使用，从而能设计出规模更小、下载更快的网页。CSS 是一套开放性标准，不仅可用于 HTML 语言，也可用于其他网页设计语言，如 XML 语言。目前 CSS 的版本包括 CSS1、CSS2 和 CSS3。

所谓“层叠”，实际上就是将显示样式独立于显示的内容，进行分类管理，如分为字体样式、颜色样式等，需要使用样式的 HTML 文件进行套用即可。CSS 标准中重新定义了 HTML 语言中原来的文字显示样式，并增加了一些新概念，如类、层等，可以对文字重叠、定位等提供更为丰富多彩的样式；同时 CSS 可进行集中样式管理。CSS 还允许将样式定义单独存储于样式文件中，这样把要显示的内容和显示样式的定义分离开，便于多个 HTML 文件共享样式定义。另外，一个 HTML 文件也可以引用多个 CSS 样式文件中的样式定义。

4.1　样式表的定义和引用

样式表的作用是通知浏览器如何呈现文档，样式表的定义是 CSS 的基础。先来看一个使用 CSS 样式定义 HTML 文件的例子。

【例 4-1】下面是一个使用 CSS 对文字显示特性进行控制的 HTML 文件。

```
<html><head><title>CSS 示例</title>
<meta http-equiv="Content-Type" content="text/html; charset=gb2312">
<style type="text/css">
h1 {font-family:"隶书", "宋体";color:#ff8800}
.text {font-family: "宋体"; font-size: 14pt; color: red}
</style></head>
<body topmargin=4><h1>这是一个 CSS 示例！</h1>
<span class="text">这行文字应是红色的。</span></body> </html>
```

本例在浏览器中的显示结果如图 4-1 所示。

在该例的头部，使用了一个新的标记<style>，这是 CSS 对样式进行集中管理的方法。在<style>标记中定义了 h1 对象的样式和一个类选择器.text，在 body 中<h1>、</h1>间的文字的显示套用 h1 对象的样式，而、之间的文字因为定义了其类名为 text，所以其显示套用类选择器.text 定义的样式。

图 4-1　CSS 样式的文字显示控制

4.1.1　样式表定义

CSS 样式表定义的基本语法为：

```
选择符（Selector）{ 规则（Rule）表}
```

其中：

（1）选择符是指要引用样式的对象，它可以是一个或多个 HTML 标记（各个标记之间以逗号分开），如例 4-1 中的 h1；也可以是类选择符（如例 4-1 中的.text）、ID 选择符或上下文选择符。

（2）规则表是由一个或多个样式属性组成的样式规则，各个样式属性间由分号隔开，每个样式属性的定义格式为：

```
样式名：值
```

样式定义中可以加入注解，格式为：

```
/*字符串*/
```

例如，font-family: "宋体"、color:red 等。以下是样式定义表的例子。

```
① p { font-family: "宋体";
    color:darkblue;
    background-color:yellow;
    font-size:9pt;      /*字体大小*/
      }
② h1,h2 { font-family:"隶书", "宋体";
    color:#ff8800;
    text-align:center;
    }
```

例①定义了一个样式表供 HTML 文件的<p>标记使用，而例②也定义了一个样式表供 HTML 文件的<h1>和<h2>标记使用。

在例②中，选择符由两个 HTML 标记组成，表示两种对象均遵循该样式定义。通常可以把描述同一个对象的样式集中在一起定义，如例①；当对象的样式很多时，也可以按照样式的类别分开定义。如例①也可定义为：

```
p { font-family: "宋体"; font-size:9pt;}
p { color:darkblue; background-color:yellow;}
```

4.1.2 样式引用

在 HTML 文件中，样式引用的方式主要有以下 4 种。

（1）链接到外部样式表

如果多个 HTML 文件要共享样式表（这些页面的显示特性相同或十分接近），则可将样式表定义为一个独立的CSS样式文件，使用该样式表的HTML文件在头部用<link>标记链接到这个CSS样式文件即可。例 4-2 给出了这种方式的用法。

【例 4-2】先将样式定义存放于文件 style.css（CSS 样式文件的扩展名为.css），style.css 文件包含的内容为：

```
h1 {font-family:"隶书","宋体";color:#ff8800}
p {background-color:yellow;color:#000000}
.text {font-family: "宋体"; font-size: 14pt; color: red}
```

HTML 文件 css1.htm 要引用该样式表，其文件内容为：

```
<html><head><title>链接外部 CSS 文件示例</title>
<link rel=stylesheet type="text/css" href="style.css" media=screen></head>
<body topmargin=4 >
<h1>这是一个链接外部 CSS 文件的示例！</h1>
<span class="text">这行文字应是红色的。</span>
<p>这一段的底色应是黄色。</p></body></html>
```

通过浏览器看到的结果如图 4-2 所示。

注意，CSS 样式文件不包含<style>标记，因为它是 HTML 标记，而不是 CSS 样式。

在 HTML 文件头部使用多个<link>标记就可以链接到多个外部样式表。<link>标记的属性主要有 REL、HREF、TYPE、MEDIA。REL 属性定义链接的文件和 HTML 文档之间的关系，通常取值为 stylesheet。HREF 属性指出 CSS 样式文件。TYPE 属性指出样式的类别，通常取值为 text/css。

MEDIA 属性指定接收样式表的显示终端，默认值为 screen（显示器），还可以是 print（打印机）、projection（投影机）等。

图 4-2　链接外部样式表文件示例

（2）引入外部样式表

这种方式在 HTML 文件的头部<style>、</style>标记之间，利用 CSS 的@import 声明引入外部样式表。格式为：

```
<style>
    @import URL("外部样式文件名");
    ...
</style>
```

例如：

```
<style type= "text/css ">
 <!--
 @import URL("style.css");
 @import URL("http://www.njim.edu.cn/style.css ");
 -->
</style>
```

引入外部样式表方式（简称引入方式）与链接到外部样式表（简称链接方式）很相似，都是将样式定义单独保存为文件，在需要使用的 HTML 文件中进行说明。两者的本质区别在于：引入方式在浏览器下载 HTML 文件时就将样式文件的全部内容复制到@import 关键字所在位置，以替换该关键字；而链接方式在浏览器下载 HTML 文件时并不进行替换，而仅在 HTML 文件体部需引用 CSS 样式文件的某个样式时，浏览器才链接样式文件，读取需要的内容。

（3）嵌入样式表

这种方式利用<style>标记将样式表嵌入 HTML 文件的头部。例 4-1 就使用了这种方式。

<style>标记内定义的前后加上注释符<!--…-->的作用是使不支持 CSS 的浏览器忽略样式表定义。<style>标记的属性 type，指明样式的类别，因为对显示样式的定义标准，除了有 CSS 外，还有 Netscape 的 JSS（JavaScript Style Sheets），其样式类别为 type=“ text/javascript ”。type 的默认值为 text/css。嵌入样式表的作用范围是本 HTML 文件。

（4）内联样式

这种方式是在 HTML 标记中引用样式定义，方法是将标记的 style 属性值赋为所定义的样式规则。由于样式是在标记内部使用的，故称为“ 内联样式 ”。

例如：

```
<h1 style="font-family:'隶书', '宋体';color:#ff8800">这是一个 CSS 示例！</h1>
<p style= "color:red;background-color:yellow ">...</p>
<body style= "font-family: '宋体';font-size:12pt;background:yellow ">
```

此时，样式定义的作用范围仅限于此标记范围之内。style 样式定义可以和原 HTML 属性一起

使用。例如：

```
<body topmargin=4 style="font-family: '宋体';font-size:12pt;background:yellow">
```

style 属性是随 CSS 扩展出来的，它可以应用于除 basefont、script、param 之外的体部标记。还要注意，若要在一个 HTML 文件中使用内联样式，必须在该文件的头部对整个文档进行单独的样式表语言声明，即

```
<meta http-equiv= "Content-type " content= "text/css ">
```

内联样式主要用于样式仅适用于单个页面元素的情况。因为它将样式和要展示的内容混在一起，自然会失去样式表的优点，表现在样式定义和内容不能分离，所以这种方式应尽量少用。

上述 4 种方式还可以混合使用，见例 4-3。

【例 4-3】设有两个样式表文件 s1.css、s2.css 和一个 HTML 文件 example_css.htm，内容分别如下。本例在浏览器中的显示效果如图 4-3 所示。

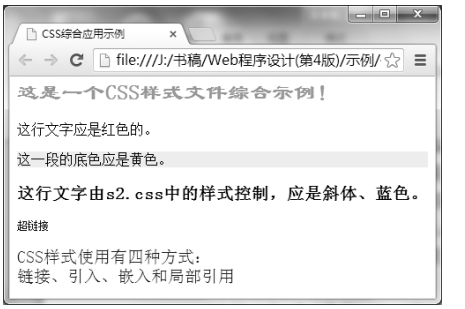


图 4-3　CSS 样式的引用方式示例

文件 s1.css：

```
h2 {font-family:"隶书";color:#ff8800}
p {color:black;background-color:yellow;font-size:12pt;}
```

文件 s2.css：

```
h3 {font-family:"宋体";color:blue;font-style:italic;}
.text {font-family: "宋体"; font-size: 10pt; color: red}
```

文件 example_css.htm：

```
<html><head><title>CSS 综合应用示例</title>
<link rel=stylesheet type="text/css" href="s1.css">
<style type="text/css">
a:visited {color: #0000FF; text-decoration: none}
a:link {font-family: "宋体"; font-size: 9pt; color: #0000FF; text-decoration: none}
a:hover {font-family: "宋体"; font-size: 12pt; color: #003333;
        background-color: #FFCC99; text-decoration: none}
@import URL("s2.css");
</style></head>
<body topmargin=4 >
<h2>这是一个 CSS 样式文件综合示例！</h2>
<span class="text">这行文字应是红色的。</span>
<p>这一段的底色应是黄色。</p>
<h3>这行文字由 s2.css 中的样式控制，应是斜体、蓝色。</h3>
<a href="a.htm">超链接</a><br><br>
<div style="font-size:14pt;color:darkred;">CSS 样式使用有四种方式：<br>
链接、引入、嵌入和局部引用</div></body></html>
```

本例样式定义中的 a:link、a:visited、a:hover 分别定义超链接在未被访问、已访问和鼠标位于超链接敏感区时的特性。

4.2　相关标记和属性

随着 CSS 的出现，有几个新的 HTML 标记和属性被增加到 HTML 语言中，以使样式表与 HTML 文件更容易地组合起来，它们是：类选择符和 class 属性、id 选择符和 id 属性、上下文选择符、伪类、span 标记和 div 标记。

4.2.1　类选择符和 class 属性

在样式引用的四种方式中，除了内联方式外，其余三种方式下，样式表中的样式定义在整个页面范围内都有效。但有时在页面中可能不希望同一种标记都遵循同一种样式，或者希望不同的标记能够遵循相同的样式。利用类选择符和标记的 class 属性就可做到这两点。方法是：在<style>标记中定义类选择符，在体部标记中将标记的 class 属性值设置为类名。

类选择符（Class Selector）在样式表中定义具有样式值的类，它有两种定义格式：

① 标记名.类名 {规则 1; 规则 2; …}

② 类名 {规则 1; 规则 2; …}

格式①的类选择符指明所定义的样式只能用在特定的标记上。例如：

```
<head><style type="text/css">
p.back { background-color:#666666;}
  …
</span>...</head>
<body>...
<p class="back">本段文字的底色为#ddeeff</p>
<p>这是另一段</p>
...</body>
```

本例定义了一个类 back 的样式，供 HTML 文件的<p>标记使用，即只有 class 属性为“ back ”的标记<p>才遵循此样式。本例<body>部分有两个<p>标记，第一个设置了 class 属性值为 back，而第二个未设置，所以只有第一个<p>标记所辖的内容遵循该样式，第二个则不遵循。

例 4-3 已经使用了格式②的类选择符，其中定义了类 text，注意这相当于*.text，标记名是用通配符表示的，匹配所有标记，即所有 class 属性值为 text 的标记都遵循此格式。这种类选择符可以使不同的标记遵循相同的样式，只要将标记的 class 属性值设置为类名即可。

4.2.2　id 选择符和 id 属性

id 选择符（ID Selector）定义一个元素独有的样式。它与类选择符的区别在于，id 选择符在一个 HTML 文件中只能引用一次，而类选择符可以多次引用。id 选择符的定义格式为：

```
#id 名 { 规则 1; 规则 2; ...}
```

要引用 id 选择符定义的样式，需在体部标记中将该 id 属性值设置为 id 名。例如：

```
<html><head>...
<style type="text/css">
...
#colorid1 { color:green;}
...</style></head>
<body>...
<h2 id="colorid1">id 选择符与 id 属性结合使用可对特定标记进行样式控制
</h2>...
</body></html>
```

当一个样式只需要在任何文档中应用一次时，使用 id 选择符是很合适的。前面已经提到，内联样式也适用此场合。两者相比，使用 id 选择符更好，因为它可以将样式定义和引用分开，并且可以应用于多个 HTML 文件。因此建议使用 id 选择符方式，尽量少用内联样式。

4.2.3　伪类

伪类是特殊的类, 可区别标记的不同状态, 能自动地被支持 CSS 的浏览器所识别。例如, visited

links（已访问的链接）和 active links（可激活链接）描述了两个锚（anchors）的状态。

伪类定义格式为：

```
选择符:伪类 { 属性: 值 }
```

伪类不用 HTML 语言的 class 属性来指定。

伪类的一个最常见的应用是指定超链接（<a>）以不同的方式显示链接（links）、已访问链接（visited links）和可激活链接（active links）。例如：

```
a:visited {color: #0000FF; text-decoration: none}
a:link {font-family: "宋体"; font-size: 9pt; color: #0000FF;
        text-decoration: none}
a:hover {font-family: "宋体"; font-size: 12pt; color: #003333;
          background-color: #FFCC99; text-decoration: none}
```

本例的含义在例 4-3 中已经分析过了，这里的 link、visited、active 都不能为 class 属性赋值，故称为伪类。

4.2.4 span 标记

标记是随 CSS 的产生而被新加入到 HTML 语言中的，增加该标记的目的是允许我们给出样式而不必将样式附加在一个 HTML 的原有标记（称为结构元素）上。它的存在纯粹是为了应用样式，所以当样式表失效时它就没有任何作用了。标记可以带有 class、id、style 等与 CSS 样式有关的属性。

4.2.5 div 标记

是 HTML 3.2 版就有的标记，是一个块级元素。 和 之间可以包含段落、标题、表格等其他块级元素。** **将其中包含的内容形成一个独立段落。 在 HTML 3.2 中只有属性 align，HTML 4.0 新增了 class、id、style 属性。 与的功能基本相同，区别在于 是块级元素，而是行元素；另外， 可包含，反之则不行。 的例子如下：

```
<div style="font-family: '宋体'; color:green; ">
<h1>DIV 标记</h1>
<p>DIV 标记在 HTML3.2 中就有定义，但只有 align 属性，
    在 HTML4.0 中增加了 class，id 和 style 属性</p>
<p>因为 DIV 可以包括其他块级元素，所以利用它可以建立复杂的文档。</p>
</div>
```

4.3 样式的继承和作用顺序

4.3.1 样式的继承

看下面的例子。

```
<html><head><title>样式继承</title>
<style type="text/css">
<!- -
h2 { color:red;}
-->
</style></head>
<body><h2><u>DIV</u></strong>标记的作用</h2></body></html>
```

在<style>标记中定义了<h2>标记的样式，在<body>中<u>、</u>标记被包含在<h2>、</h2>中，那么<u>标记是否引用<h2>的样式呢？回答是肯定的。这就是样式继承的概念：我们将包含其他标记的标记称为父标记，则被包含的标记就是子标记，子标记将继承父标记的样式。在本例中包含在<u>和</u>之间的文字“ DIV ”将显示为红色。

样式的继承还有一种特殊形式——相对值继承方式，即以百分比继承。例如：

```
<style>
p.class1 {font-size:12pt;}
p.class2 {font-size:200%}
p.class3 {font-size:100%}
</style>
```

若在 body 部分有以下语句：

```
<p class="class1">第一段</p>
<p class="class2">第二段</p>
<p class="class3">第三段</p>
```

图 4-4　样式的相对值继承示例

则在浏览器中的显示效果如图 4-4 所示。

本例中的 p.class2 和 p.class3 样式的 font-size 属性分别以 200%、100%的比例继承 p.class1 的 font-size 属性值，即两者的 font-size 值分别为 200%×12pt=24pt，100%×12pt=12pt。

4.3.2　样式的作用顺序

样式的作用域指对一个标记究竟哪个样式起作用。提出这个问题的原因在于，对一个标记来说可能有多个样式都符合生效条件。例如：

```
<html><head><title>样式的作用顺序</title>
<style type="text/css">
    p { color:red; font-size:22pt;}
    p.c1{color:green; font-size:12pt; }
    p {font-size:16pt;text-align:center;}
</style></head>
<body><p style="color:#ffaa66">第一段</p>
<p class="c1">第二段</p><p>第三段</p></body></html>
```

在这个例子中，针对<p>标记定义了三个样式表，对<p>中的文字和布局方式进行了说明。body 部分共出现了三个<p>标记，它们分别应该应用哪个样式表呢？

样式表的作用优先顺序遵循以下四条原则。

（1）内联样式中所定义的样式优先级最高。

（2）其他样式表按其在 HTML 文件中出现或被引用的顺序，越在后出现，优先级越高。

（3）选择符的作用顺序由高到低为：上下文选择符、类选择符、id 选择符。

（4）未在任何文件中定义的样式，将遵循浏览器的默认样式。

依据这些原则，对上例进行分析。第一和第三个样式表定义了 color、text-align、font-size，两个表中都有 font-size 属性，显然只有后一个值生效；所以对不带 class 和 style 属性的<p>标记，套用的样式值为 color:red，text-size:16pt，text-align:center。第二个样式表从属于类选择器 p.c1，只有 class 属性为 c1 的<p>标记才能引用，注意在这个样式表中只定义了 color 和 font-size，所以在其他<p>样式表中定义的 text-align 样式值，对 class 属性为 c1 的<p>标记也会生效。再来看 body 中的第一个<p>标记，它使用了内联样式，该内联样式仅定义了 color 属性，那么该<p>标记的其他显示属性将遵循样式表定义或使用浏览器默认样式，因此该<p>标记中的内容的显示属性值应为 color:#ffaa66，font-size:16pt，text-align:center，其余显示属性为浏览器默认值。同样可以分析出第二、三个<p>标记中内容的显示属性值分别应为 color:green,red；font-size:12pt,16pt；

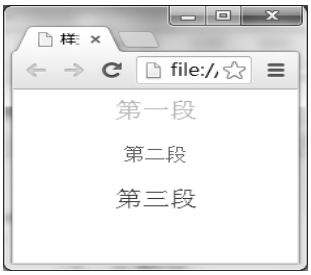


图 4-5　样式的作用顺序示例

text-align:center,center；其余显示属性为浏览器默认值，参见图 4-5。

由上例可以看出，当同时引用多个样式文件时，样式表的作用顺序较复杂，应特别注意。如果希望一个属性的值不被其他样式定义中相同属性的定义所覆盖，可用特定参数!important。例如将前例中第一个<p>样式定义改为：

```
p { color:red; font-size:22pt !important;}
```

则浏览器显示的“第一段”、“第二段”和“第三段”的字号都将为 22pt。

4.4　CSS 属性

CSS1 属性可分为字体属性、颜色和背景属性、文本属性、方框属性、分类属性和定位属性等几部分。本节将讨论每类属性的概况、常用属性的含义和用法。

4.4.1　字体属性

字体属性包括字体（font-family）、字号（font-size）、字体风格（font-style）、字体加粗（font-weight）、字体变化（font-variant）及字体综合设置（font）等属性。字体属性的含义明确，使用简单，下面用一示例说明其用法。

【例 4-4】CSS 字体属性用法示例。它在浏览器中的显示结果如图 4-6 所示。

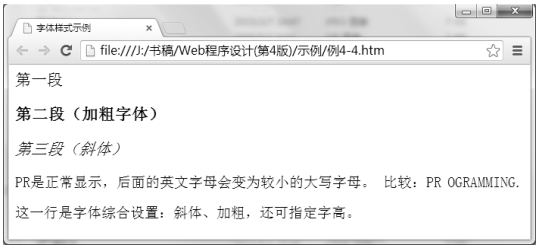


图 4-6　CSS 的字体属性示例

```
<html><head><title>字体样式示例</title>
<style type="text/css">
body {font-family:"宋体","隶书";}
p {font-size:16pt;}
p.weight_1{font-weight:100;}
p.weight_9{font-weight:900;}
p.font_i{font-style:italic;}
span {font-size:14pt;}
span.font_n {font-variant:normal;}
span.font_v {font-variant:small-caps;}
span.font_all {font: bold italic 30px/40px;}
</style></head>
<body><p class="weight_1">第一段</p>
<p class="weight_9">第二段（加粗字体）</p>
<p class="font_i">第三段（斜体）</p>
<span class="font_n">PR 是正常显示，后面的英文字母会变为较小的大写字母。
        比较：PR</span>
<span class="font_v">OGRAMMING.</span><br><br>
<span class="font_all">这一行是字体综合设置：斜体、加粗，还可指定字高。</span>
</body></html>
```

4.4.2　颜色和背景属性

颜色属性允许设计者指定页面元素的颜色，背景属性指定页面的背景颜色或背景图像的属性。颜色和背景类属性包括（前景）颜色（color）、背景颜色（background-color）、背景图像（background-image）、背景重复（background-repeat）、背景附属方式（background-attachment）、背景图像位置（background-position）及背景属性（background）。表 4-1 列出了常用的颜色和背景属性。

表 4-1　颜色和背景属性表

属　性　名	可　取　值	含　　义	举　　例
color	英文单词 #RRGGBB #RGB	指定页面元素的前景色	h1{color:red} h2{color:#008800} h3{color:#080}
background-color	英文单词 #RRGGBB #RGB transparent	指定页面元素的背景色	body {background-color:white} h1{background-color:#0000F0} p { background-color:transparent}
background-image	统一资源定位器 URLs none	指定页面元素的背景图像	body {background-image:url(bg.gif)} p { background-image: url(http://www.htmlhelp.com/bg.jpg)}
background-repeat	repeat repeat-x repeat-y no-repeat	决定一个被指定的背景图像被重复的方式。默认值为 repeat	body {background-repeat:no-repeat} p {background-repeat:repeat-x}
background-attachment	scroll fixed	指定背景图像是否跟随页面内容滚动。默认值为 scroll	body {background-attachment:fixed}
background-position	数值表示法 关键词表示法	指定背景图像的位置	body {background-position:30% 70%} p {background-position:bottom left}
background	背景颜色、背景图像、背景重复、背景位置	背景属性综合设定	body {background:url(bg1.gif) green repeat-y fixed left 20pt}

背景图像位置（background-position）属性可以确定背景图像的绝对位置，这是 HTML 标记不具备的功能。该属性只能应用于块级元素和替换元素（包括 img、input、textarea、select、object）。background-position 值的表示有两种方式：数值表示法和关键词表示法。

数值表示法用坐标值表示位置，坐标原点是背景图像位置属性所属元素的左上角。数值表示法又分为百分比表示和长度值表示两种。百分比表示的格式为：X%　Y%；长度值表示的格式为：Xpt Ypt。它们的含义如图 4-7 所示。

例如，值 100pt 40pt 表示指定图像会被放于其所属元素的左起 100pt、上起 40pt 的位置。

关键词表示法以相应的英文单

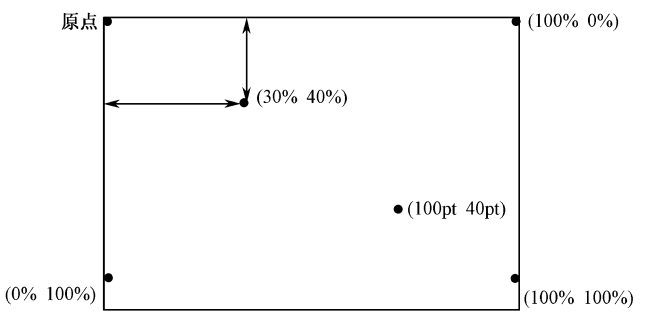


图 4-7　background-position 属性值表示

词表示位置。横向关键词有 left、center、right，纵向关键词有 top、center、bottom。关键词含义解释如下：

```
top left=left top=0% 0%
top=top cnter=center top=50% 0%
top right=right top=100% 0%
left=left center=center left=0% 50%
center=center center=50% 50%
right=right center=center right=100% 50%
bottom left=left bottom=0% 100%
bottom=bottom center=center bottom=50% 100%
bottom right=right bottom=100% 100%
```

百分比和长度值的两种数值表示方法可以混用，如 30% 10pt，但不能和关键词表示法混用。长度表示法中如果只指定一个值，那么该值作为横向值，垂直值则默认为 50%。例如，background-position:30%与 background-position:30% 50%相同。

【例 4-5】CSS 颜色和背景属性的用法示例。它在浏览器中的显示结果如图 4-8 所示。

```
<html><head><title>颜色和背景属性的使用</title><style>
body {background-image:url(bg1.gif);background-repeat:repeat-y;}
p {color:green;background-color:aqua;background-image:url(bg2.gif);
background-repeat:no-repeat; background-position:40% 40pt}
</style></head>
<body><p>这是一段文字<br>
本段有一不同于 body 的背景图<br>
它从 40% 40pt 处开始显示<br>
并且不重复<br><br><br></p><br><br>
背景属性也可以用在 style 属性中，例如：<br>
<table width=90% border=2 cellpadding=50 cellspacing=2>
<tr><td style="color:darkred;text-align:right;background-repeat:no-repeat;
background-image:url(bg3.jpg);background-position:bottom left">
<span>本格背景图在[0% 100%]处</span></td>
<td style="color:red;background-repeat:no-repeat;
    background-image:url(bg3.jpg);background-position:top right">
<span>本格背景图在[100% 0%]处</span>
</td></tr><table></body></html>
```

图 4-8　颜色和背景属性的用法示例

4.4.3　文本属性

文本属性设置文字之间的显示特性，包括字符间隔（letter-spacing）、文本修饰（text-decoration）、大小写转换（text-transform）、文本横向排列（text-align）、文本纵向排列（vertical-align）、文本缩排（text-indent）、行高（line-height）。现将文本属性的属性名、可取值及相关说明列于表 4-2 中。

表 4-2　文本属性表

属　性　名	可　取　值	含　　义	举　　例
letter-spacing	长度值|normal	设定字符之间的间距	h1 {letter-spacing:8pt} p {letter-spacing:14pt}
text-decoration	none|underline| overline| line-through| blink	设定文本的修饰效果，line-through 是删除线，blink 是闪烁效果。默认值为 none	a:link,a:visited,a:active { text- decoration:none}
text-align	left|right|center| justify（将文字均分展开对齐）	设置文本横向排列对齐方式	p {text-align:center} h1 {text-align:right}
vertical-align	baseline|super| sub|top|middle| bottom|text-top| text-bottom| 百分比	设定元素纵向对齐方式。值的含义见下面的说明。默认值为 baseline	img.mid { vertical-align:50%} span.sup { vertical-align:super} span.sub { vertical-align:sub}
text-indent	长度值|百分比	设定块级元素第一行的缩进量	p { text-indent:30pt} h1 { text-indent:10%}
line-height	normal|长度值| 数字|百分比	设定相邻两行的间距。默认值 normal	p { line-height:200%} p { line-height:30pt}

说明：

① vertical-align 属性的默认值为 baseline，表示该元素与其上级元素的基线对齐；该属性的值为百分比，表示在其上级元素的基线上变化的比例。该属性的其他值的含义如下：

super：上标；　　sub：下标；

top：垂直向上对齐；　　middle：垂直居中对齐；

bottom：垂直向下对齐；　　text-top：文字向上对齐；

text-bottom：文字向下对齐。

② line-height 属性的默认值为 normal，表示由浏览器自动调整行间距；该属性的值为数字，表示行间距等于文字大小乘以该数字所得的数值；该属性值为百分比，表示行间距为字大小的百分比。例如，字的大小为 14pt，line-height 属性值为 200%，则行间距为 14pt×200%=28pt。

【例 4-6】CSS 文本属性的用法示例。

```
<html><head><title>文本属性用法</title>
<style type="text/css">
h2.space {letter-spacing:10pt;}
p.ind {text-indent:20pt;color:darkred;background-color:#FFAAAA}
h3.dec {text-decoration:line-through}
p.hei1{line-height:16pt}
p.hei2 {line-height:32pt}
span.super {vertical-align:super;}</style></head>
<body>
<h2 class="space">本行字符间距是 10pt</h2>
```

```
<p class="ind">本段文字起始缩进 20pt，<br>然后可以跟正文。</p>
<h3 class="dec">本行文字带有删除线。</h3>
<p class="hei1">本行与下一行间距为 16pt，<br>本行与上一行间距为 16pt。</p>
<p class="hei2">本行与下一行间距为 32pt，<br>本行与上一行间距为 32pt。</p>
本行的 X 和 Y 带有上标：X<span class="super">3</span>+Y
<span class="super">3</span></body></html>
```

例 4-6 在浏览器中的显示结果如图 4-9 所示。

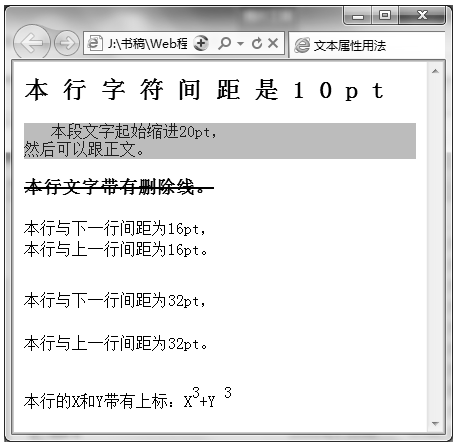


图 4-9　CSS 的文本属性用法示例

4.4.4　方框属性

方框属性用于设置元素的边界、边框等属性值，可应用这些属性的元素大多是块元素，包括 body、p、div、td、table、h*x*（*x*=1,2, …,7）等。方框属性包括边界（margin）、边界补白（padding）、边框（border）等的设置，这部分属性繁多，设置方法复杂，详见附录 C（“ CSS 样式表属性 ”）。以下通过例 4-7 说明常用的方框属性的用法。

【例 4-7】CSS 方框属性用法示例。

```
<html><head><title>方框属性用法例子</title>
<style type="text/css">
p { background-color:#ddeeff;
margin-top:10;          margin-right:30;
margin-bottom:10;     margin-left:30;
border-width:20pt;     border-style:groove;
border-color:blue;      padding:20pt;
width:600;                height:350;
}
img.float{ float:left;}
img.nofloat {clear:both}
</style></head>
<body><table border=1><tr><td>
<p>这是本段的开始文字！本行距段边框 20pt。<br><br>
<img class="float" src="hua.gif">这些文字应该围绕在图像右边显示。
左边这幅图像是一束花。<br>
<img class="nofloat" src="img2.jpg"><br>这些文字不围绕在图像两边。
</p></td></tr></table></body></html>
```

例 4-7 中，定义了标记<p>的方框属性：它的上下边框距其上级元素（本例中是表<table>）的边界距离为 10，左右边框距其上级元素的边界距离为 30；边框宽度为 20pt；边框的样式是 3D 凹

线；边框颜色为 blue；<p>中内容距边框的距离为 20pt；<p>元素的宽度和高度分别是 500、300。本例还设置了元素与其周围文字的显示特性，定义了一个浮动文字的类 img.float 和一个清除浮动文字的类 img.nofloat。例 4-7 在浏览器中的显示结果如图 4-10 所示。

图 4-10　CSS 方框属性用法示例

4.4.5　列表属性

列表属性用于设置列表标记（ol 和 ul）的显示特性，包括 list-style-type、list-style-image、list-style-position、list-style 等属性，它们的名称、含义和相应说明列于表 4-3 中。

【例 4-8】CSS 列表属性用法示例。

```
<html><head><title>列表属性用法</title>
<style type="text/css">
ul.ul1 { list-style:square inside;}
ul.ul2 { list-style-image:url("check.gif");
         list-style-position:outside;}
ol.ol1 { list-style-type:upper-roman;
         list-style-position:inside;}
ol.ol2 { list-style:decimal outside;}
</style>
    </head>
    <body><h3>计算机系</h3>
    <ul class="ul1">
          <li>计算机及应用 99（1）班</li>
          <li>计算机及应用 99（2）班</li>
    </ul>
    <ul class="ul2">
          <li>计算机及应用 99（3）班</li>
          <li>计算机及应用 98（1）班</li>
    </ul>
    <h3>电子系</h3>
    <ol class="ol1">
          <li>电子信息工程 99（1）班</li>
          <li>电子信息工程 99（2）班</li>
    </ol>
    <ol class="ol2">
          <li>电子信息工程 98（1）班</li>
```

```
        <li>电子信息工程 98（2）班</li>
    </ol>
    </body></html>
```

表 4-3　列表属性表

属　性　名	取　　值	含　　义
list-style-type	无序列表值： disc|circle|square 有序列表值： decimal| ower-roman| upper-roman| lower-alpha| upper-alpha 共用值：none	表项的项目符号。disc—实心圆点；circle—空心圆；square—实心方形；decimal—阿拉伯数字；lower-roman—小写罗马数字；upper-roman—大写罗马数字；lower-alpha—小写英文字母；upper-alpha—大写英文字母；none—不设定
list-style-image	url（URL）	使用图像作为项目符号
list-style-position	outside|inside	设置项目符号是否在文字里，与文字对齐
list-style	项目符号，位置	综合设置项目属性

4.4.6　定位属性

CSS 提供用于二维和三维空间定位的属性，它们是 top、left、position。利用它们可以将元素定位于相对其他元素的相对位置或绝对位置。

1．top、left、position 属性

top 属性设置元素与窗口上端的距离；left 属性设置元素与窗口左端的距离；position 属性设置元素位置的模式。top 和 left 属性通常配合 position 属性使用。

position 有如下三种取值。

（1）absolute：绝对位置，原点在所属块元素的左上角。

（2）relative：相对位置，该位置是相对 HTML 文件中本元素的前一个元素的位置。

（3）static：静态位置，按照 HTML 文件中各元素的先后顺序显示。

position 的默认值为 static。

【例 4-9】CSS 二维定位属性用法示例。本例在浏览器中的显示结果如图 4-11 所示。

```
<html><head><title>二维定位属性用法</title>
<style type="text/css">
p { font-size:12pt; color:green; }
div.block1 { position:absolute; top:80; left:120;
             width:200;height:200;
            background-color:#ddeeff; }
img.pos1 { position:relative; top:20; left:20;
             width:80;height:80; }
div.block2 { position:absolute; top:80; left:420;
             width:200;height:200;
            background-color:#ddeeff; }
img.pos2 { position:absolute; top:20; left:20;
             width:80;height:80; }
</style></head>
<body><div class="block1"><img class="pos1" src="img1.gif"><br>
```

```
<p>这是一幅鲜花图像。</p></div>
<div class="block2"><img class="pos2" src="img1.gif"><br>
<p>这是一幅鲜花图像。</p></div>
</body></html>
```

图 4-11　CSS 的二维定位属性用法示例

2．三维空间定位

CSS 允许在三维的空间中定位元素，与之相关的属性是 z-index，z-index 与 top 和 left 属性结合使用。z-index 将页面中的元素分成多个“层”，形成多个层“堆叠”效果，从而营造出三维空间效果。

z-index 的取值为整数，可以为正，也可为负，值越大表示在堆叠层中越处于高层，为 0 表示基准，为负表示位置在 z-index=0 的元素之下。

【例 4-10】CSS 三维空间定位属性用法示例。本例在浏览器中的显示结果如图 4-12 所示。

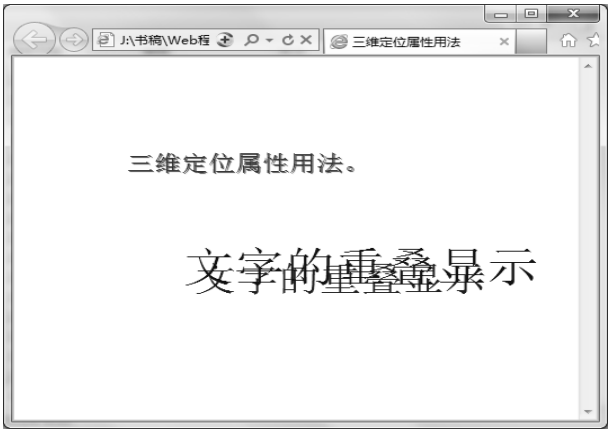


图 4-12　CSS 三维定位属性用法示例

```
<html><head><title>三维定位属性用法</title>
<style type="text/css">
span { font-size:18pt;}
span.level2 { position:absolute; z-index:2; left:100;top:100; color:red;}
span.level1 { position:absolute; z-index:1; left:101;top:101; color:green;}
span.level0 { position:absolute; z-index:0; left:102;top:102; color:yellow;}
```

```
p.lev1 { position:absolute; top:200;left:150; z-index:2; font-size:34pt;color:blue;}
p.lev2 { position:relative; top:202 ;left:150; z-index:-2;
        font-size:28pt;color:darkred; }</style></head>
<body><span class="level2">三维定位属性用法。</span>
<span class="level1">三维定位属性用法。</span>
<span class="level0">三维定位属性用法。</span>
<p class="lev1">文字的重叠显示</p><p class="lev2">文字的重叠显示</p>
</body></html>
```

4.5 CSS+DIV 页面布局

在网页设计中，网页布局最基本的要求是，要考虑浏览者的方便程度并能够明确地传达信息，以及兼顾网页设计的审美，给浏览者一定的视觉享受。网页布局就是把网页的各种构成要素，如文字、图像、图标、菜单等，合理地排列起来。以前常使用表格来对页面进行布局，随着互联网与 Web 技术的发展，Web 标准的网页布局已经成为以后 Web 的发展方向。当前，使用 CSS（层叠样式表）+DIV（层）对页面进行排版布局已成为标准的方式。使用 CSS+DIV 的布局模式使页面具有易于维护、显示效果好、浏览器兼容性好、下载速度快、适应不同终端需要等优点。

CSS+DIV 页面布局的核心在于使网页达到表现与内容的分离，即网站的结构、表现、行为三者分离。只有真正实现了结构分离的网页设计，才是真正意义上符合 Web 标准的网页设计。有关 CSS+DIV 页面布局的具体技术细节，请读者参考有关资料。

4.6 应用实例——设计个人主页

本节通过个人主页实例对 CSS 样式的使用进行总结，读者可从例子中得到启迪，多做多练，以达到举一反三、灵活运用的目的。

【例 4-11】设计如图 4-13 所示的个人主页。该主页使用表格作为主要结构，一个表的表项又是另一个表。表结构在页面设计中应用非常广泛，它可以灵活方便地规划显示区域。在 Internet 上许多 Web 页面都是应用表结构设计的，它还有下载速度快的优点。

图 4-13　个人主页示例

本例大量使用了样式表，在头部通过<style>标记集中定义了页面的显示样式，通过内联样式定义了页面按钮风格的栏目“ 团结 ”、“ 进取 ”等表项，使得页面显示风格灵活多样。

例 4-11 源代码如下：

```
<html>
<meta http-equiv=Content-Type content="text/html; charset=gb2312">
<style type=text/css>
    a:link {font-size: 9pt; text-decoration: none}
    a:visited {font-size: 9pt; text-decoration: none}
    a:active {font-size: 9pt; text-decoration: none}
    a:hover {color: red; text-decoration: none}
    body {font-size: 9pt; line-height: 14pt}    table {font-size: 9pt; line-height: 14pt}
    tr {font-size: 9pt }                        td {font-size: 9pt }
    .e {font-size: 16pt; font-family: "MS Sans Serif"; text-decoration: none}
</style></head>
<body bgColor=#ffffff leftMargin=0 background="fallb.jpg" topMargin=0>
<div align=center><center>
<table cellSpacing=0 cellPadding=0 width=720 border=0>
    <tr><td width="100%"><div align=center><center>
        <table cellPadding=0 width=760 border=0>
          <tr><td width="27%"><p align=center>
              <img border="0" src="s1.gif" width="200" height="40"></p></td>
              <td width="73%"></td></tr></table></center></div>
        <div align=center><center>
        <table cellPadding=2 width=743 border=0 name="nav">
         <tr><td style="border-right: 1px ridge; border-top: 1px ridge; border-left: 1px ridge;
                border-bottom: 1px ridge" align=middle width=103 bgColor=#a7d6ba>团结</td>
            <td style="border-right: 1px ridge; border-top: 1px ridge; border-left: 1px ridge;
                border-bottom: 1px ridge" align=middle width=103 bgColor=#a7d6ba>进取</td>
            <td style="border-right: 1px ridge; border-top: 1px ridge; border-left: 1px ridge;
                border-bottom: 1px ridge" align=middle width=103 bgColor=#a7d6ba>友谊</td>
            <td style="border-right: 1px ridge; border-top: 1px ridge; border-left: 1px ridge;
                border-bottom: 1px ridge" align=middle width=103 bgColor=#a7d6ba>开朗</td>
            <td style="border-right: 1px ridge; border-top: 1px ridge; border-left: 1px ridge;
                border-bottom: 1px ridge" align=middle width=103 bgColor=#a7d6ba>奋斗</td>
            <td style="border-right: 1px ridge; border-top: 1px ridge; border-left: 1px ridge;
                border-bottom: 1px ridge" align=middle width=103 bgColor=#a7d6ba>成功</td>
            <td style="border-right: 1px ridge; border-top: 1px ridge; border-left: 1px ridge;
                border-bottom: 1px ridge" align=middle width=86 bgColor=#a7d6ba>主页</td>
        </tr></table></center></div>
        <div align=center><center>
        <table cellSpacing=0 cellPadding=0 width=720 border=0>
        <tr><td width=718 bgColor=#ffffff colspan=2><hr color=#abd1ef size=5></td></tr>
        <tr><td width=105 bgColor=#ffffff rowspan=3>
              <img border="0" src="web.png" width="105" height="360"></td>
            <td width=613><p align=center>
              <font color=#ff6c26><span class=e>◆</span></font>
              <font color=#008000 size="3">精彩人生</font>
              <font color=#ff6c26><span class=e>◆</span></font></p></td></tr>
             <tr><td width=613><div align=center><center>
               <table cellspacing=0 cellPadding=0 width="85%" border=0>
               <tr><td width="100%">
               <table borderColor=#ffffff cellPadding=2 width="100%" border=1>
```

```
                    <tr><td borderColor=#70b8e2 width="100%" colspan=2>
                    <img border="0" src="18.gif" width="510" height="32"></td></tr>
                    <tr><td borderColor=#70b8e2 width="23%">学生时代</td>
                    <td borderColor=#70b8e2 width="77%">连续三年当选学生班
                        干部并多次被评为三好学生</td></tr>
                    <tr><td borderColor=#70b8e2 width="23%">工作历程</td>
                    <td borderColor=#70b8e2 width="77%">工作第一年获先进个
                        人称号</td></tr>
                    <tr><td borderColor=#70b8e2 width="23%">工作历程</td>
                    <td borderColor=#70b8e2 width="77%">第二年获优秀</td></tr>
                    <tr><td borderColor=#70b8e2 width="23%">工作历程</td>
                    <td borderColor=#70b8e2 width="77%">第三年提前晋级</td></tr>
                    <tr><td borderColor=#70b8e2 width="23%">工作历程</td>
                    <td borderColor=#70b8e2 width="77%">第四年获先进个人
                        称号</td></tr>
                    <tr><td borderColor=#70b8e2 width="23%">工作历程</td>
                    <td borderColor=#70b8e2 width="77%">第五年获先进个人
                        称号</td></tr>
                    </tr></table></center></div></td></tr></table></center></div>
    </td></tr></table></center></div>
</body>
</html>
```

本 章 小 结

本章主要讨论了使用层叠样式表进行页面设计的技术。CSS 是 W3C 协会制定的一套扩展样式标准，其中重新定义了 HTML 中原来的文字显示样式，并增加了一些新概念，如类、层等，可以对文字重叠、定位等，提供了更为丰富多彩的样式；并且 CSS 可进行集中样式管理；此外 CSS 还允许将样式定义单独存储于样式文件中，这样可以把显示的内容和显示样式的定义分离开，便于多个 HTML 文件共享样式定义。CSS 属性可分为字体属性、颜色及背景属性、文本属性、方框属性、分类属性和定位属性等几大类。

习 题 4

4.1 简述层叠样式表 CSS 的含义和作用。

4.2 试总结 CSS 样式引用的方式，并举例说明。

4.3 试总结样式作用的顺序，并举例说明。

4.4 试总结常用的 CSS 属性。

4.5 试用 HTML、CSS 样式表设计你的个人主页，主要内容包括简介、兴趣爱好、特长等。

上机实验 4

实验 4.1 用 CSS 字体、颜色和文本等属性控制网页显示样式。

【目的】掌握用 CSS 属性控制网页文字和颜色显示样式的方法。

【内容】利用 CSS 样式表实现对文字的显示控制（文字内容可自行选择）：分别设置字体为隶

书、楷体、宋体，字号分别为 20pt、16pt 和 24pt，字体风格分别为正常、斜体、正常，字符间距分别为 2pt、5pt、2pt，第二行文字带有背景色，最后一行文字带有删除线。例如图 4-14 所示页面效果。

【步骤】

（1）打开记事本程序。

（2）输入能够生成如图 4-14 所示页面的 html 文件（图片可另选），保存为 ex4-1.html 文件。

（3）双击 ex4-1.html 文件，在浏览器中查看结果。

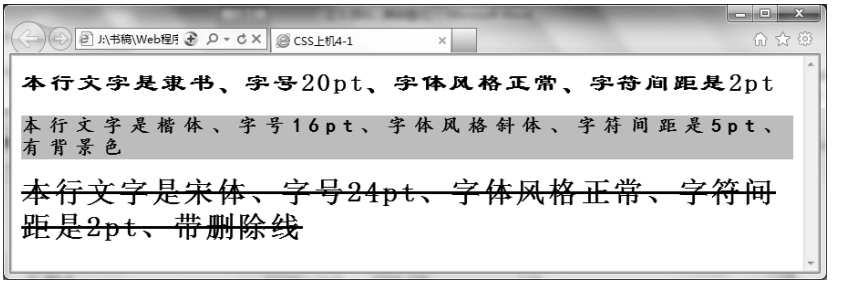


图 4-14　用 CSS 字体、颜色和文本等属性控制网页显示样式

实验 4.2　用 CSS 控制网页显示样式。

【目的】掌握用 CSS 样式表控制网页显示样式的方法。

【内容】利用 CSS 样式表实现例 4-11 中的个人主页（图片可另选）。

【步骤】

（1）打开记事本程序。

（2） 输入能够生成如图 4-13 所示页面的 html 文件（图片可另选），保存为 ex4-2.html 文件。

（3）双击 ex4-2.html 文件，在浏览器中查看结果。

第 5 章　Web 客户端程序设计

网页设计要使用多种技术，包括 HTML 语言、脚本程序设计、CSS 样式表以及美工技术等。仅使用 HTML 语言设计的页面属于静态页面。Web 刚出现的一段时间内，Web 是一个静态信息发布平台，所设计的页面都是静态页面。而当今的 Web 已经具有更丰富的功能。现在，人们不仅需要浏览 Web 提供的信息，而且还需要进行信息搜索，开展电子商务等。为实现以上功能，必须使用更新的网络编程技术设计动态网页。所谓动态，指的是按照访问者的需要，对访问者输入的信息作出不同的响应，提供响应信息。更进一步，动态网页设计技术又可分为客户端和服务器端，客户端动态网页设计技术主要使用层叠样式表（CSS）和在浏览器中执行的脚本程序，而服务器端动态网页设计技术主要使用 ASP.NET、JSP、PHP 等。

随着 Web 页面的内容和表现手法越来越丰富，将其结构、表现和行为分离成为趋势。构建精良的 Web 页面一般有三个层次，分别是：结构（Structure）层、表现（Presentation）层和行为（Behavior）层。对应的标准也分三方面：结构化标准语言（主要包括(X)HTML 和 XML）、表现标准语言（主要指 CSS）和行为标准（主要包括对象模型和脚本语言）。Web 客户端程序设计的主要内容是脚本语言和浏览器对象模型，本章对它们进行讨论。

5.1　脚本语言 JavaScript

5.1.1　什么是脚本语言

脚本（Script）语言的概念源于 UNIX 操作系统，在 UNIX 操作系统中，将主要以行命令组成的命令集称为 Shell 脚本程序。Shell 脚本程序具有一定的控制结构，可以带参数，由系统解释执行。除了 UNIX Shell Script 外，在 UNIX 环境下，具有强大的字符串处理能力的 Perl 语言也是脚本语言的典型代表。

随着 Internet 的发展，特别是 WWW 应用的迅速普及，人们不再满足于静态的页面浏览，希望网页具有动态交互的特性，因此各种应用于 Web 页面设计的脚本语言应运而生。其中应用较广泛的是 JavaScript 及用于编写 CGI 脚本程序的 Perl、Shell Script 等。脚本程序设计在 Web 程序设计中占有重要的地位，无论是客户端动态页面设计，还是动态网站设计及服务器端编程，都要使用脚本语言。

HTML 语言提供较完善的设计页面的功能，但它提供的信息大多是静态的。这些信息被下载到客户计算机后，是固定不变的。无法利用客户计算机的计算能力，也就无法在客户端处理与用户的交互，从而无法构造出客户端的交互式动态页面。一些原本可以在客户端完成的任务（如数据合法性检查等）也不得不提交给服务器去完成，这一方面加重了服务器的负担，另一方面也增加了网络传输量，同时还加长了响应时间，降低了实时性。JavaScript 的出现恰好弥补了这一缺憾，它大大提高了客户端的交互性，使用非常简单、灵活。

本章所讨论的脚本语言是指用于 Web 页面及程序设计的脚本语言，它们通常是嵌入式（嵌入到 HTML 文件中）的、具有解释执行的特征。根据脚本程序被解释执行的地点的不同，可将它们分为客户端脚本和服务器端脚本，前者由浏览器负责解释执行，后者由 Web 服务器负责解释执行。JavaScript 既可作为客户端脚本语言，又可作为服务器端脚本语言，而 Perl、JSP 及 PHP 等则通常

是服务器端脚本语言。

5.1.2　JavaScript 语言概述

JavaScript 是一种嵌入在 HTML 文件中的脚本语言，它是基于对象和事件驱动的，能对诸如鼠标单击、表单输入、页面浏览等用户事件做出反应并进行处理。它由 Netscape 公司在 1995 年的 Netscape 2.0 中首次推出，最初被叫做“ Mocha ”，当在网上测试时，又将其改称为“ LiveScript ”，到 1995 年 5 月 Sun 公司正式推出 Java 语言后，Netscape 公司引进 Java 的有关概念，将 LiveScript 更名为 JavaScript。在随后的几年中，JavaScript 语言被大多数浏览器所支持。就目前使用最广泛的两种浏览器 Netscape 和 Internet Explorer 来说，Netscape 2.0 及以后的版本、IE 3.0 及以后的版本都支持 JavaScript 脚本语言，所以 JavaScript 具有通用性好的优点。

最初 JavaScript 只是作为客户端编程语言，随着发展它已经可以完成较复杂的服务器端编程任务了。JavaScript 从推出发展到今天，经过几次升级，目前版本是 1.8 版，它在网络安全性和服务器端支持方面比较早的版本有更好的支持。本节只讨论客户端使用的 JavaScript，其特点是直接由浏览器解释运行。

JavaScript 具有如下特点。

（1）简单性。JavaScript 是一种被大幅度简化了的编程语言，即使用户没有编程经验也可较快掌握它。它不像高级语言的使用有很严格的限制，而是非常简洁灵活，例如在 JavaScript 中变量可以直接使用，不必事先声明，对变量的类型规定也不是十分严格等。

（2）基于对象。JavaScript 是基于对象的，它允许用户自定义对象，同时浏览器还提供了大量内建对象，使编程者可以将浏览器中不同的元素作为对象来处理，体现了现代面向对象程序设计的基本思想。但 JavaScript 不是完全面向对象的，它不支持类和继承。

（3）可移植性。在大多数浏览器上，JavaScript 脚本程序可以不经修改而直接运行。

（4）动态性。JavaScript 是 DHTML（动态 HTML）的一个十分重要的部分，是设计交互式动态网页、特别是“ 客户端动态 ”页面的重要工具。

另外，还要说明一下 JavaScript 语言与 Java 语言的关系，这二者在命名、结构和语言上都很相似，但不能把它们混淆，两者存在如下重要的差别。

（1）Java 是由 Sun 公司推出的新一代的完全面向对象的程序设计语言，它支持类和继承，主要应用于网络程序设计，对于非程序设计人员来说不易掌握。而 JavaScript 只是基于对象的，主要用于编写网页中的脚本，易于学习和掌握。

（2）Java 程序是编译后以类的形式存放在服务器上，浏览器下载到这样的类，用 Java 虚拟机去执行它。而 JavaScript 的源代码无须编译，它是嵌入在 HTML 文件中的，作为网页的一部分。当使用能处理 JavaScript 语言的浏览器浏览该网页时，浏览器将对该网页中的 JavaScript 源代码进行识别、解释并执行。

（3）Java 程序可以单独执行，但 JavaScript 程序只能嵌入 HTML 文件中，不能单独运行。

（4）Java 具有严格的类型限制，JavaScript 则比较宽松。

（5）Java 程序的编辑、编译需要使用专门的开发工具，如 JDK（Java Development Kit）、Visual J++等。而 JavaScript 程序不需要特殊的开发环境，由于它只是作为网页的一部分嵌入到 HTML 文件中，所以编辑 JavaScript 程序只要用一般的文本编辑器就可以完成。

5.1.3　JavaScript 编程基础

1．JavaScript 程序的编辑和调试

可以用任何文本编辑器来编辑 JavaScript 程序，例如 NotePad，需要将 JavaScript 程序嵌入

HTML 文件，程序的调试在浏览器中进行。

将 JavaScript 程序嵌入 HTML 文件的方法有如下两种。

（1）在 HTML 文件中使用<script>、</script>标识加入 JavaScript 语句，这样 HTML 语句和 JavaScript 语句位于同一个文件中。其格式为：<script language="JavaScript">。

其中，language 属性指明脚本语言的类型。通常有两种脚本语言：JavaScript 和 VBScript，language 的默认值为 JavaScript。<script>标记可插入在 HTML 文件的任何位置。

（2）将 JavaScript 程序以扩展名“ .js ”单独存放，再利用以下格式的 script 标记嵌入 HTML 文件：<script src=JavaScript 文件名>。

方法（2）将 HTML 代码和 JavaScript 代码分别存放，有利于程序的共享，即多个 HTML 文件可以共用相同的 JavaScript 程序。<script>标记通常加在 HTML 文件的头部。

下面是一个简例：

```
<html>
    <head>
    <title>JavaScript 简例</title>
    </head>
    <body>
      <script language="JavaScript"> alert( "世界，你好!" );</script>
    </body>
</html>
```

图 5-1　一个 JavaScript 简例

本例在 HTML 文件中用<script>和</script>标记嵌入了一个 JavaScript 语句 alert()，它是 JavaScript 浏览器对象 Window 的预定义方法，其功能是弹出具有一个“ 确定 ”按钮的对话框。该对话框上所显示的内容为其参数所给的字符串。该例运行结果如图 5-1 所示。

另外，在编写 JavaScript 程序时还要注意以下三点。

（1）JavaScript 的大小写是敏感的，这点与 C++相似。

（2）在 JavaScript 程序中，换行符是一个完整的语句的结束标志；若要将几行代码放在一行中，则各语句间要以分号（ ;）分隔（习惯上，也可像 C++一样，在每一个语句之后以一个分号结束，虽然 JavaScript 并不要求这样做）。

（3）JavaScript 有单行、多行两种注释（与 C++相同）。插入单行注释的符号是“ // ”，插入多行注释的符号是“ /* ”和“ */ ”，以“ /* ”开始，“ */ ”结束。JavaScript 的注释不会被解释和执行。

下面讨论 JavaScript 的基本语法。JavaScript 的基本语法与 C 语言很相似，它继承了 C 语言的优点，并融入了面向对象的思想。

2．数据类型

JavaScript 有三种数据类型：数值型、逻辑型和字符型。

（1）数值型。数值型数据包括整数和浮点数。整数可以是十进制、八进制和十六进制数，八进制值以 0 开头，十六进制值以 0x 开头。例如，100（十进制），021（八进制），0x5d（十六进制）。

以下是浮点数例子：2.57，1.3e6，2，7e-10。

（2）逻辑型。逻辑型数据有 true 和 false 两种取值，分别表示逻辑真和逻辑假。

（3）字符型。字符型数据的值是以双引号或单引号括起来的任意长度的一连串字符。注意反斜杠“ \ ”是转义字符，常用的转义序列有：

\n　//换行符
\t　//水平制表符
\r　//回车符
\b　//退格符

3．常量和变量

（1）常量。常量是在程序中其值保持不变的量。JavaScript 的常量以直接量的形式出现，即在程序中直接引用值，如“ 欢迎您 ”、“ 28 ”等。

常量值可以为整型、实型、逻辑型及字符串型。另外，JavaScript 中有一个空值 null，表示什么也没有，如试图引用没有定义的变量，则返回一个 null 值。

（2）变量。变量是在程序中值可以改变的量。JavaScript 用关键字 var 声明变量，或使用赋值的形式声明变量。例如：

```
var str;                /*声明变量 str*/
num1=10;                /*说明 num1 为整型，并将其值赋为 10*/
num2=3.02e10;
str1="欢迎您";
```

JavaScript 的变量使用比较灵活，可以在程序中需要之处声明变量、为变量赋值，而不必事先将程序中要用到的所有变量都做声明；并且可以不声明而直接使用变量，如上面例子所示的那样。JavaScript 的变量使用的灵活性还体现在其弱类型检查特点上，弱类型检查允许对一个变量随时改变其数据类型。例如：

```
num1=10;                /*说明 num1 为整型*/
num1=3.02e10;           /*将 num1 改变为浮点型*/
num1="欢迎您";      /*甚至还可将 num1 改变为字符串型*/
```

JavaScript 命名变量的规则是：

① 变量名必须以字母（大小写均可）打头，只能由字母（大小写均可）、数字（0~9）和下画线“ _ ”组成；

② 变量名长度不能超过 1 行，并且不能使用 JavaScript 保留字作变量名；

③ 变量名字母区分大小写。

表 5-1 列出了 JavaScript 的保留字，保留字是系统预先定义，具有特殊含义和用途的字符串，不能作他用。

表 5-1　JavaScript 的保留字

abstract	boolean	break	byte	case	catch
char	class	const	continue	default	do
double	else	extends	false	final	finally
float	for	function	goto	if	implements
import	in	instanceof	int	interface	long
native	new	null	package	private	protected
public	return	short	static	super	switch
synchronized	this	throw	throws	transient	true
try	var	void	while	with	

4．运算符和表达式

（1）运算符

JavaScript 的运算符包括赋值运算符、算术运算符、字符串运算符、逻辑运算符、关系运算符和位运算符。

① 赋值运算符。JavaScript 提供 6 个赋值运算符，它们是基本赋值运算符“ = ”，复合赋值运算符：“ += ”、“ -= ”、“ *= ”、“ /= ”和“ %= ”，功能是将一个表达式的值赋予一个变量。复合赋值运算符的含义见表 5-2。例如：

```
x=100;          //将值 100 赋予变量 x
a+=10;          //即 a←a+10
```

表 5-2　赋值运算符简记形式表

记　法	含　义	记　法	含　义
a+=b	a=a+b	a-=b	a=a-b
a*=b	a=a*b	a/=b	a=a/b
a%=b	a=a%b	a<<b	a=a<<b
a>>=b	a=a>>b	a>>>=b	a=a>>>b
a&=b	a=a&b	a^=b	a=a^b
a|=b	a=a|b		

② 算术运算符。算术运算符的操作数和结果都是数值型值。JavaScript 的算术运算符列于表 5-3 中。算术运算符及后面要讲的位运算符可与赋值运算符结合形成简记形式，如表 5-3 所示。

表 5-3　算术运算符表

运　算　符	操　作	运　算　符	操　作
+	加法	-（双目）	减法
*	乘法	/	除法
%	取模	++	递增
- -	递减	-（单目）	取负

③ 字符串运算符。字符串运算是 JavaScript 中使用最多的运算。字符串运算符只有一个“ + ”，即字符串连接运算。参与字符串连接运算的两个操作数如果都是字符串，则直接合并；否则，操作数会先被转变为字符串，再进行合并。例如：

```
var str1="欢迎您"+"访问本页";      //变量 str1 的值为“ 欢迎您访问本页 ”
var str2="现在是"+10+"月";         //变量 str2 的值为“ 现在是 10 月 ”
```

④ 逻辑运算符。逻辑运算符的运算对象和结果都是逻辑值。逻辑运算符有如下三个。

&&　与运算，是双目运算。当两个操作数都为 true 时，结果为 true，其他情况下结果均为 false。

||　或运算，是双目运算。当两个操作数中至少有一个为 true 时，结果为 true，否则结果为 false。

!　非运算，是单目运算。结果是操作数的值取反。

⑤ 关系运算符。关系运算符用于数值及字符串值的比较，返回比较判断的结果。关系运算符的运算结果是逻辑值。关系运算包括：

==　相等　　!=　不等　　<　小于

>　大于　　<=　小于或等于　　>=　大于或等于

例如：*x*>=100，*y*==20。

利用关系运算符、逻辑运算符及括号可以组成复杂的表达式。例如：

```
(！(a==9）&&（x<=100））‖ （a!=9））
```

⑥ 位运算符。位运算符将操作数作为二进制值处理，返回 JavaScript 标准的数值型数据。位运算符都是双目运算，包括：

& 按位与	| 按位或	^ 按位异或
<< 左移	>> 右移	>>> 右移，零填充

例如：15&8 的结果为 8（1111&1000）；

15|8 的结果为 15（1111|1000）；

15^8 的结果为 7（1111|1000）。

JavaScript 运算符的优先级由高到低排列如下：

```
[ ]  ()                          高
++   --   !
*   /   %   +   -
<<   >>   >>>
<   >   <=   >=
==   !=
&   ^   |
&&   ‖
?=
=   +=   -=   *=   /=   %=       低
```

（2）表达式

JavaScript 的表达式是由常量、变量、运算符、函数和表达式组成的式子，任何表达式都可求得单一值。根据表达式值的类型，JavaScript 的表达式有如下三类。

① 算术表达式。其值是一个数值型值。例如：5+*a*-*x*。

② 字符串表达式。其值是一个字符串。例如："字符串 1"+str。

③ 逻辑表达式。其值是一个逻辑值。例如：(x= =y) && (y>=5)。

此外，JavaScript 还有一种特殊的表达式——条件表达式，其格式为：

```
(condition) ? val1 : val2
```

其中，condition 是逻辑表达式。该条件表达式的含义是：如果 condition 的值为 true，则条件表达式的值为 val1，否则条件表达式的值为 val2。

例如：((date>20) && (date<30))? "go" : "stay"，该表达式的含义是，若变量 date 的值大于 20 且小于 30，则表达式的值为字符串 go，否则为 stay。

5．函数

函数为程序设计人员提供了实现模块化的工具。通常在进行一个复杂的程序设计时，总是根据所要完成的功能，将程序划分为一些相对独立的部分，每部分编写一个函数，从而使各部分充分独立，任务单一，程序清晰，易懂、易读、易维护。JavaScript 函数可以封装那些在程序中可能要多次用到的功能块。函数定义的语法格式为：

```
return 表达式
```

或

```
return （表达式）
```

下例说明函数的定义和调用方法。

【例 5-1】设计一个如图 5-2 所示的页面，显示指定数的阶乘值。

程序如下：

```
<html><head><title>函数简例</title>
    <script language="JavaScript">
        function factor(num){
        var i,fact=1;
        for (i=1;i<num+1;i++)
            fact=i*fact;
        return fact;
        }
    </script>
</head>
<body><p><script>
        document.write("<br><br>调用 factor()函数，5 的阶乘等于：",factor(5),"。");
</script></p>
</body></html>
```

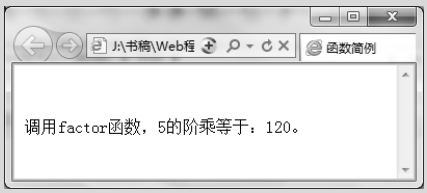


图 5-2　函数使用简例

例 5-1 在 HTML 文件头部定义了函数 factor(num)，它计算 num!并返回该值；在 HTML 体部的脚本程序中调用了 factor，实参值为 5，即计算 5!。

使用函数时要注意以下三点。

（1）函数定义位置。虽然语法上允许在 HTML 文件的任意位置定义和调用函数，但建议在 HTML 文件的头部定义所有的函数，因为这样可以保证函数的定义先于其调用语句载入浏览器，从而不会出现调用函数时由于函数定义尚未载入浏览器而引起的函数未定义错误。

（2）函数的参数。函数的参数是在主调程序与被调用函数之间传递数据的主要手段。在定义函数时，可以给出一个或多个形式参数，而在调用函数时，却不一定要给出同样多的实参。这是 JavaScript 在处理参数传递上的特殊性。JavaScript 中，系统变量 arguments.length 中保存了调用者给出的实在参数的个数。例 5-2 给出了函数参数传递的用法。

【例 5-2】设计一个函数求累加和，默认时求 1+2+…+1000，否则按照用户所指定的开始值和终止值求和。在浏览器中执行的结果如图 5-3 所示。

```
<html><body><script language= "JavaScript">
function sum(StartVal,EndVal)
{ var ArgNum = sum.arguments.length;   //用户给出的参数个数
   var i,s=0;
   if (ArgNum == 0 )
   {   StartVal = 1; EndVal = 1000; }
   else if (ArgNum == 1 )
            EndVal = 1000;
   for (i = StartVal; i<=EndVal; i++)
           s+=i;
   return s;
}
document.write("不给出参数调用函数 sum:",sum(),"<br>");
document.write("给出一个参数调用函数 sum:",sum(500),"<br>");
document.write("给出二个参数调用函数 sum:",sum(1,50),"<br>");
</script>
</body></html>
```

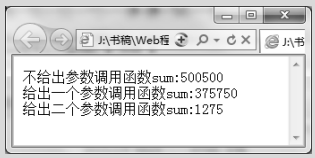


图 5-3　带参函数的使用

（3）变量的作用域。在函数内用 var 保留字声明的变量是局部变量，其作用域仅局限于该函数；而在函数外用 var 保留字声明的变量是全局变量，其作用域是整个 HTML 文件。在函数内未用 var 声明的变量也是全局变量，其作用域是整个 HTML 文件。当函数内以 var 声明的变量与全局变量同名时，它们就像不同名的两个变量，其操作互不影响。有关变量作用域见例 5-3。

【例 5-3】变量作用域示例。执行结果如图 5-4 所示。

```
<html><head><title>变量作用域示例</title>
<script language="JavaScript">
   var i, j=10;     //全局变量
   function output(){
       var j=0;     //局部变量
       i=100;       //全局变量
       j++; j++;
       document.write(" j=",j);
       document.write(" i=",i);
       i++;
   }
   </script>
</head>
<body><br><br>
<script language= "JavaScript">
    document.write("尚未调用函数 output()，所以 i 无定义，不能引用！<br>");
    document.write("j 的初始值=",j,"<br>");
    document.write("调用 output()，观察函数的输出！<br>");
    output();
    document.write("<br>调用 output()后，观察函数对 i,j 的影响：i=",i," j=",j);
</script></body></html>
```

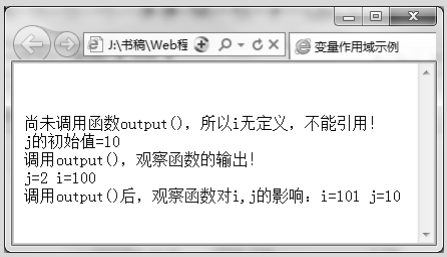


图 5-4　变量作用域示例

6．流程控制

在任何一种语言中，程序控制流都是必需的，它能使得整个程序顺利地按一定的方式执行。与所有其他的程序设计语言一样，JavaScript 有顺序、分支和循环三种控制结构。顺序结构是最一般的控制结构，若没有改变执行顺序的语句，则程序的各语句是按其出现的先后顺序依次执行的。可以改变程序执行顺序的是条件转移语句和循环语句。

（1）条件转移语句

条件转移语句定义的语法格式为：

```
if (condition)   statments1
     [else   statments2 ]
```

其中，condition 表示条件，可以是逻辑或关系表达式，若是数值型数据，则将零和非零的数分别转换成 false 和 true。如果 condition 为 true，则执行语句体 statments1；若省略 else 子句，则 condition 为 false 时什么也不做，否则执行语句体 statments2。

若 if 及 else 后的语句体有多行，则必须使用花括号将其括起来。

if 语句可以嵌套，格式为：

```
if（condition1）statments1
    else if（condition2）statments2
    else if（condition3）statments3
      ...
    else statments N；
```

在这种情况下，每一级的条件表达式都会被计算，若为真，则执行其相应的语句，否则执行 else 后的语句。

（2）while 循环语句

while 循环语句定义的语法格式为：

```
while (condition) {statments}
```

当 condition 为 true 时，反复执行循环体 statements；否则，跳出循环体。要注意在循环体中必须含有改变循环条件的操作，使之离循环终止更近一步，否则会陷入死循环。

（3）for 循环语句

for 循环语句定义的语法格式为：

```
for (exp1；exp2；exp3) {statments}
```

其中，exp1 是循环前的初始设置，通常设置循环计数器的初值；exp2 是循环条件，当 exp2 为 true 时才执行循环体 statments；exp3 是运算，它改变循环设置，通常会改变循环计数器的值，使之离循环终止更近一步。

for 与 while 两种语句都是循环语句，它们的表达能力是相当的。但习惯上当使用循环计数器进行控制时选用 for 语句，因为在这种情况下用 for 语句更为清晰、易读，也较紧凑；而 while 语句对循环条件较复杂的情况更加适合。

【例 5-4】使用 for 循环语句计算 10！。

```
<html><body>
<script language= "JavaScript">
    var i,factor;
    factor=1;
    for (i=1;i<=10;i++) factor*=i;
    document.write("10 的阶乘是：",factor);
 </script>
</body></html>
```

（4）continue 和 break 语句

continue 语句强制本轮循环结束，进入下一轮循环。例如：

```
while (i<100){
    if (j==0) continue;
    else {语句体}
    j++;
}
```

上例中，如果 j 为 0，则本轮循环结束（语句 j++不执行）。

break 语句强制结束循环。例如：

```
while (i<100){
    if (j==0) break;
    else {语句体}
    j++;
}
```

上例中，如果 j 为 0，则 while 循环结束。

7．事件触发和处理

JavaScript 是基于对象（Object-based）的语言，而基于对象的基本特征，就是采用事件驱动（Event-driven）。HTML 文件中的 JavaScript 应用程序通常是事件驱动程序，事件（Events）是指对计算机进行一定的操作而得到的结果，例如将鼠标移到某个超链接上、按下鼠标按钮等都是事件。由鼠标或热键引发的一连串程序的动作，称为事件驱动（Event Driver）。对事件进行处理的程序或函数，称为事件处理程序（Event Handler）。JavaScript 定义了常用事件的名称、何时及何对象发生此事件（即事件触发）以及事件处理名，表 5-4 列出了几个最常用的事件及相应的事件处理名。

表 5-4　JavaScript 常用事件表

事　件　名	发生的对象	说　　明	事件处理名
Click()	表单的 button，radio，checkbox，submit，reset，link（超链接）	单击了表单元素或超链接	onClick
Load()	HTML 的 body 元素	在浏览器中载入页面	onLoad
Unload()	HTML 的 body 元素	退出当前页面	onUnload
MouseOver()	link	鼠标移到超链接上	onMouseOver
MouseOut()	link	鼠标移出超链接	onMouseOut
Submit()	form	用户提交了表单	onSubmit

有关事件触发与处理的编程还与浏览器对象密切相关，将在第 5.2 节详细讨论，这里先举一个例子。

【例 5-5】MouseOver()和 MouseOut()事件处理用法示例。

```
<html><head><title>事件触发和事件处理</title>
  <script language="JavaScript">
        var Images=new Array();
        Images[0]=new Image(); Images[0].src="dot1.jpg";
        Images[1]=new Image(); Images[1].src="check.gif";
        function changeImg(ImgIndex) {   document.imgs.src=Images[ImgIndex].src; }
  </script></head>
<body>
<center><a href="learn.html" onMouseOver="changeImg(1); return true"
onMouseOut="changeImg(0); return true">
<img src="dot1.jpg" name="imgs" border=0 width=30 height=30><font size=5>
软件设计</font> </a></center>
</body></html>
```

例 5-5 定义了一个含有两个元素的全局数组 Images，函数 changeImg()根据参数值为 body 中的 name 属性值为 imgs 的图像（Img）元素的 src 属性赋值。例中超链接设置 MouseOver()和 MouseOut()的事件处理程序以不同的参数值（分别为 1 和 0）调用 changeImg()函数，执行结果为当鼠标位于该超链接时，imgs 图像的 src 属性值被赋为 check.gif（即在该图像元素位置上显示 dot1.jpg），当鼠标离开该超链接时，imgs 图像的 src 属性值被赋为 dot1.jpg。图 5-5 所示左边画面为鼠标离开超链接时的显示，右边画面为鼠标位于超链接时的显示。本例中应用了 Document 对象的子对象 Images 的 src 属性，该属性对应 img 标记的 src 属性。

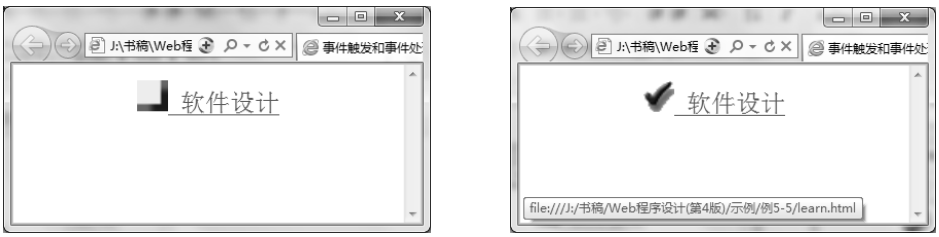


图 5-5　MouseOver()和 MouseOut()事件处理示例

8．综合举例——一个简易计算器的设计

这部分综合运用 JavaScript 的基本语法知识，设计一个较为复杂的 JavaScript 程序——一个基于 Web 的简易计算器。

【例 5-6】简易计算器设计。所谓简易计算器，就是只能进行加、减、乘、除 4 种运算，且仅

进行简单的正确性检查——只检查除数是否为零。程序运行的结果如图 5-6 所示。

图 5-6　简易计算器程序的运行结果

首先，需要设置数字按键和功能按键，可使用 HTML 表单按钮（button）来表示。例如，使用如下语句：

```
<input type=button value="1" onClick="SetVal('1')">
```

显示数字“ 1 ”的按键，当按下该按键时，将执行 SetVal('1')操作。

```
<input type=button value="+" onClick="SetOpr('+') ">
```

显示运算符“ + ”的按键，当按下该按键时，将执行 SetOpr('+')操作。

又如以下语句：

```
<input type=button value="=" onClick="Compute(this.form) ">
```

显示功能按键“ = ”，当按下该键时，将计算用户输入的表达式的值。

其次，需要一个显示输入计算式和结果的地方，可使用 HTML 表单的 text（单行文本框）元素来表示，例如：

```
<input type=text value=" " name=OutText>
```

最后，要考虑这些设置和计算任务如何来完成。

（1）SetVal 操作：将用户按下的键所代表的数字连接到整个输入串的尾部，并判断这是第几个操作数，将其存入相应的变量中；

（2）SetOpr 操作：将用户按下的键所代表的运算连接到整个输入串的尾部；

（3）Compute 操作：利用系统预定义函数 eval()求出表达式的值；

（4）Clear 操作：清除输入框的内容。

以下是例 5-6 源程序清单：

```
<html><head><script language="JavaScript">
<!--
//定义全局变量
var n1='',n2='';                          //定义两个变量，分别存放两个操作数
var item1_flag=true;                      //标志是否第一个操作数
var opr_type='+';                         //运算类型
function SetVal(item){                    //在输出框中置数值
   document.Cal.OutText.value+=item;      //字符串连接
   if (item1_flag)                        //若是第一个操作数
       n1+=item;                          //将其加入变量 n1
   else
       n2+=item;
}
function SetOpr(opr){                     //在输出框中置运算符
   document.Cal.OutText.value+=opr;
   item1_flag=false;
   opr_type=opr;
}
function Clear(){                         //清除输出框的内容
   document.Cal.OutText.value="";
   item1_flag=true;      opr_type='+';   n1=" ";   n2=" ";
}
function Compute(obj){                    //计算表达式的值
```

```
    var Result;
    if ((n1!='') && (n2!='')){
    if ((eval(n2)==0) && (opr_type=='/'))
    {   alert('除数不能是 0!');
        Clear();
        return;
    }
    else
    {     Result=eval(obj.OutText.value);
          document.Cal.OutText.value+='=';
          document.Cal.OutText.value+=Result;
       }
    }
}
//-->
</script></head><body><p align=center><form name="Cal" >
<input type="text" value="" name="OutText"><br><br>
<input type="button" value=" 0 "    onClick="SetVal('0')">
<input type="button" value=" 1 "    onClick="SetVal('1')">
<input type="button" value=" 2 "    onClick="SetVal('2')">
<input type="button" value=" 3 "    onClick="SetVal('3')"><br><br>
<input type="button" value=" 4 "    onClick="SetVal('4')">
<input type="button" value=" 5 "    onClick="SetVal('5')">
<input type="button" value=" 6 "    onClick="SetVal('6')">
<input type="button" value=" 7 "    onClick="SetVal('7')"><br><br>
<input type="button" value=" 8 "    onClick="SetVal('8')">
<input type="button" value=" 9 "    onClick="SetVal('9')">
<input type="button" value=" + "    onClick="SetOpr('+')">
<input type="button" value=" - "    onClick="SetOpr('-')"><br><br>
<input type="button" value=" * "    onClick="SetOpr('*')">
<input type="button" value=" / "    onClick="SetOpr('/')">
<input type="button" value=" CE "    onClick="Clear()">
<input type="button" value=" = " onClick="Compute(this.form)">
</form></p></body></html>
```

5.1.4　JavaScript 对象

前面已经提到，JavaScript 语言是基于对象的，在 JavaScript 中，对象是对客观事物或事物之间的关系的刻画。JavaScript 的对象有内建对象和用户自定义对象两大类，内建对象包含了对浏览器各成分的描述，是 JavaScript 程序设计中应用最多的部分；用户自定义对象允许用户根据需要创建自己的对象，从而进一步扩大 JavaScript 的应用范围，增强编写功能强大的 Web 文档。

1．JavaScript 对象概述

JavaScript 中的对象是由属性（Properties）和方法（Methods）两个基本元素构成的：属性成员是对象的数据；方法成员是对数据的操作。

要使用一个对象，可采用以下三种方式。

（1）引用 JavaScript 内建对象。

（2）由浏览器环境提供，即引用浏览器对象。

（3）创建自定义对象。

要注意的是：一个对象在被引用之前，这个对象必须存在，否则将出现错误。实际上，引用对象要么创建新的对象，要么利用现存的对象。

2．自定义对象

这里介绍自定义对象的创建方法。

用户定义自己的对象包括构造对象的属性和定义对象的方法两部分，下面通过例子来说明对象的定义方法。

【例 5-7】“ 书 ”对象的定义。

```
function print()
{//方法成员定义，输出各属性成员值
    document.write("书名为"+this.name+"<br>");
    document.write("作者为"+this.author+"<br>");
    document.write("出版社为"+this.publisher+"<br>");
    document.write("出版时间为"+this.date+"<br>");
    document.write("印数为"+this.num+"<br>");
}
function book(name,author,publisher,date,num)
{//构造函数
    this.name=name;              //书名，属性成员
    this.author=author;          //作者，属性成员
    this.publisher=publisher;    //出版社，属性成员
    this.date=date;              //出版时间，属性成员
    this.num=num;                //印数，属性成员
    this.print=print;            //方法成员
}
```

例 5-7 定义了“ 书 ”对象，book()是该对象的构造函数，该对象有 5 个属性成员：name、author、publisher、date 和 num，有一个方法成员 print()，作用是输出对象的属性值。

从例 5-7 可以看出，定义一个对象的步骤是：首先定义对象的各个方法成员，每个方法成员就是一个普通函数，然后定义对象的构造函数，其中包含每个属性成员的定义和初始化，以及每个方法成员的初始化。

构造函数从形式上看与普通函数相同，但有如下特殊性。

（1）构造函数的名字就是对象的名字。如例 5-7 所定义的对象的名字就是构造函数 book 的名字——book。

（2）在构造函数中常使用关键字 this 来为对象的属性成员和方法成员初始化，this 本身是一个特殊对象，即当前构造函数正在创建的对象。

（3）每个对象都必须定义构造函数。

3．对象的引用

要引用对象，必须先用保留字 new 创建对象的实例。JavaScript 中，对象是对具有相同特性的实体的抽象描述，而对象实例则是具有这些特性的单个实体。

创建对象实例的方法是：

```
var 对象实例名=new 对象名(实在参数表);
```

创建对象实例时，要注意实在参数表与对象构造函数的形式参数表的对应关系。

例如：对例 5-7 定义的 book 对象创建实例。

```
var book1=new book("语文","集体编","人民教育出版社","1999",10000);
```

创建了对象实例后，就可通过该实例引用对象的属性和方法成员。

对象属性成员的引用格式是：

```
对象实例名.属性成员名
```

对象方法成员的引用格式是：

```
对象实例名.方法成员名
```

例如：

```
book-name=book1.name;
book1.print();
```

还需说明的是，从概念上严格区分时，对象和对象实例的含义是不同的，但通常为叙述简洁，在不会引起误解之处，本书也将对象实例简称为对象，读者可从上下文判断其含义。

4．有关对象操作的语句

JavaScript 提供了两个用于操作对象的语句。

（1）for…in 语句。这是一条循环语句，格式如下：

```
for（变量名 in 对象实例名）
```

该语句用于对已有对象实例的所有属性进行操作的控制循环，它将一个对象实例的所有属性反复置给指定的变量来实现循环，而不是使用计数器来实现。该语句的优点就是无须知道对象中属性的个数即可进行操作。

【例 5-8】下列函数 Show()显示其参数对象各属性的值，它可作为一个通用函数使用。

```
<html><body><script language= "JavaScript">
function person(name,age)   //定义对象 person
{   this.name=name;
    this.age=age;
}
function book(title,author,publisher,price)   //定义对象 book
{   this.title=title;
    this.author=author;
    this.publisher=publisher;
    this.price=price;
}
function Show(obj)   //定义通用函数 Show
{    var prop;
     for (prop in obj)
     document.write(obj[prop]+"    ");
     document.write("<br>");
}
var obj1=new person("Mary",20);
var obj2=new book("语文","集体编","人民教育出版社",5.5);
Show(obj1);
Show(obj2);
</script></body></html>
```

调用函数 Show()时，在循环体中，for 自动将其属性取出来，直到最后，不需要知道对象属性的个数。

若不使用 for…in 语句，就要通过数组下标值来访问每个对象的属性，使用这种方式时首先必须知道对象属性的个数，否则若超出范围，就会发生错误；而且对于不同的对象要进行不同的处理，因为各对象的属性成员数一般不相同。而通过数组下标值访问对象属性的方法在新版本的浏览器中已不被支持。

例 5-8 使用了另一种访问对象属性的方法：

```
对象实例名[属性成员名]
```

例如：book1("title")。

这种引用对象属性的方式通常只用在 for…in 语句中。

（2）with 语句。从前述可以了解到，当需要引用对象的属性或方法成员时，都要在成员名前缀上对象的名字。例如，对于对象 book1 的实例 b1，若要引用其成员，则要使用如下的格式：

```
book1.title
book1.author
book1.publisher
book1.date
book1.num
book1.print()
```

为了简化书写，可使用 with 语句，其语法格式是：

```
with object{
  //在其中引用 object 的成员时，可不加前缀
}
```

使用该语句的意思是：在该语句体内，任何对变量的引用被认为是这个对象的属性，以节省一些代码。例如：

```
with book1 {
    document.write(title);      //实际上是引用 book1 对象的 title 属性
    document.write(author);
    ...
}
```

此外，在 JavaScript 中，可以向已定义的对象中增加属性。通常，定义一个对象，在定义了其构造函数后，该对象的数据结构就已经确定了；如果要向该对象中加入数据，即要改变对象的数据结构，此时不需重新设计构造函数，可以通过构造函数的 prototype 属性来添加新的属性成员。例如，若想在例 5-7 中定义的 book 对象中添加一个属性 price，可使用如下的语句：

```
book.prototype.price=10;
```

这样，在该语句之前或之后创建的所有对象实例都会具有 price 属性成员，并且该属性的值为 10。

5.1.5 常用的内建对象和函数

5.1.4 节讨论了对象的基本概念及自定义对象的定义和使用方法，而在 JavaScript 中，掌握和使用 JavaScript 的预定义对象（即内建对象）和浏览器对象才是最重要的。5.2 节将详细讨论浏览器对象模型。下面介绍 JavaScript 的内建对象。

JavaScript 提供的一些非常有用的常用内建对象和方法如下。

（1）Array（数组）对象。JavaScript 的数组可通过该内建对象来实现。

（2）String（字符串）对象。封装了字符串及有关操作。

（3）Math（数学）对象。封装了一些常用的数学运算。

（4）Date（日期时间）对象。封装了对日期和时间的操作。

（5）Number 对象、Boolean 对象、Function 对象。

另外还有一些常用的预定义函数。对这些预定义函数，JavaScript 并未用对象来封装它们，故不能把它们归于对象中。本节将介绍这些对象和函数，它们为编程人员快速开发强大的脚本程序提供了有效手段。

1．数组

数组是若干元素的有序集合，每个数组有一个名字作为其标识。在几乎所有的高级语言中，数组都是得到支持的数据类型，但在 JavaScript 中，没有明显的数组类型。在 JavaScript 中数组可通过对象来实现，具体有两种实现方式：① 使用 JavaScript 的内建对象 Array；② 使用自定义对象的方式创建数组对象。

（1）内建对象 Array

① 创建数组对象实例。通过 new 保留字来进行，其语法格式如下：

```
var  数组名=new Array([数组长度值]);
```

其中，数组名是一个标识符。数组长度值是一个正整数。

例如：

```
var arr1=new Array();          //创建数组实例 arr1，长度不定
var arr2=new Array(10);              //创建数组实例 arr2，长度为 10
```

若创建数组时不给出元素个数，则数组的大小由后面引用数组时确定。数组的下标从 0 开始，因此有 10 个元素的数组，其下标范围是 0～9。

② 数组元素的引用。引用数组元素的语法格式为：

```
数组名[下标值]
```

例如：

```
arr1[2]              //定义数组 arr1，大小为 2
arr2[6]              //定义数组 arr2，大小为 6
```

③ 内建对象 Array 的特点。Array 的使用较灵活，在以下两点上与大多数高级语言不同：

- 数组元素不要求数据类型相同

例如，可以给一个数组的不同元素赋予不同类型的值：

```
arr1[0]=10;                  //数值型
arr1[1]= "王林";             //字符串
arr1[2]=false;               //逻辑型
```

另外，数组的元素还可以是对象。当数组元素是数组对象时，就得到一个二维数组。例如：

```
var arr=new Array(10);
for (i=0;i<10;i++)
      arr[i]=new Array(5);
```

这样就创建了一个 10×5 的二维数组。

二维数组元素的引用方法为：

```
数组名[第一维下标值][第二维下标值]
```

例如：arr[2][3]。

- 数组长度可以动态变化

例如，前面定义了有 10 个元素的数组 arr2，若希望增加到 18 个元素，则只要用以下赋值语句即可：

```
arr2 [17]=1;              //可以为 arr2[17]赋任意值
```

④ Array 对象的属性和方法。

Array 对象常用的属性是 length 属性，表示数组长度，其值等于数组元素个数。

其常用方法有：

join()　　该方法返回由数组中所有元素连接而成的字符串；

reverse()　　该方法逆转数组中各元素，即将第一个元素换为最后一个，将最后一个元素换为第一个；

sort()　　对数组中的元素进行排序。

【例 5-9】一个 Array 对象的应用示例。

```
<html><head><title>数组对象</title>
<script language="JavaScript">
function updateInfo(WhichBook)   //对象 book 的方法成员，修改对象属性值
{ document.BookForm.currbook.value=WhichBook;
  document.BookForm.BookTitle.value=this.Title;
   document.BookForm.BookPublisher.value=this.Publisher;
  document.BookForm.BookAmount.value=this.Amount;
}
function Book(title,publisher,amount)   //对象 book 的构造函数
{ this.Title=title;
  this.Publisher=publisher;
  this.Amount=amount;
  this.UpdateInfo=updateInfo;
}
</script></head>
<body><script language="JavaScript">
var Books=new Array(); //创建数组，数组元素是 book 对象
//为数组各元素赋值
Books[0]=new Book("语文","少年儿童出版社",10000);
Books[1]=new Book("数学","高等教育出版社",5000);
Books[2]=new Book("普通物理","高等教育出版社",3000);
Books[3]=new Book("计算机基础","清华大学出版社",2000);
</script>
<h2 align=center>共有四本书，可选择查看其信息</h2>
<form name="BookForm">
选择当前所显示的书：  
<input type=button value=A 书 onClick="Books[0].UpdateInfo('A 书')">
<input type=button value=B 书 onClick="Books[1].UpdateInfo('B 书')">
<input type=button value=C 书 onClick="Books[2].UpdateInfo('C 书')">
<input type=button value=D 书 onClick="Books[3].UpdateInfo('D 书')"><br><br>
当前书：<input type="text" name="currbook" value="A 书"><br><br>
书名：<input type="text" name="BookTitle" value="语文"><br><br>
出版社：<input type="text" name="BookPublisher" value="少年儿童出版社"><br><br>
印数：<input type="text" name="BookAmount" value="10000"></form></body></html>
```

例 5-9 的功能是：按照用户单击的按钮（A 书、B 书、C 书或 D 书），分别在“当前书”、“书名”、“出版社”和“印数”框中显示相应的书代号、书名、出版社名和印数，其运行结果如图 5-7 和图 5-8 所示。

例 5-9 在 HTML 文件头部的脚本部分定义了一个对象 book，它有三个属性成员 title、publisher 和 amount，及一个方法成员 updateInfo，作用是将表单对象相应的域赋予指定的值。在体部的脚本部分创建了 book 对象数组 Books，它有 4 个元素，每个元素都是一个 book 对象实例。该 HTML 文件的 body 部分是生成一个表单，共有 4 个文本框，分别显示当前书的代号、书名、出版社和数量；它有 4 个按钮，分别代表 4 本书的选择。要注意定义这 4 个按钮时 onClick 事件处理的设置。

例如：

```
<input type=button value=D 书 onClick="Books[3].UpdateInfo('D 书')">
```

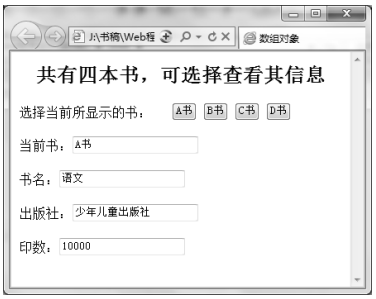


图 5-7　例 5-9 的初始显示

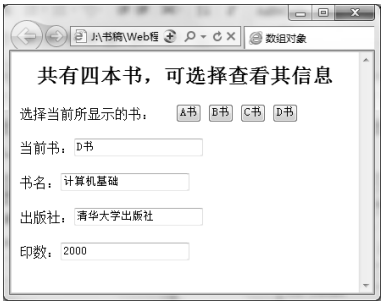


图 5-8　例 5-9 选择“ D 书 ”后的显示

表示若单击代表“ D 书 ”的按钮，则执行 Books[3].UpdateInfo('D 书')，即在 4 个文本框中分别显示 D 书的信息。

（2）自定义数组对象

除了直接使用 JavaScript 的 Array 对象实现数组外，由于数组是一个对象，所以也可以像自定义对象那样实现数组。在早期的 JavaScript 版本中甚至并未提供 Array 预定义对象。自定义数组对象与一般的自定义对象的使用方法一样：通过 function 定义一个数组的构造函数，并使用 new 对象操作符创建一个具有指定长度的数组。

① 定义数组对象。

```
function arrayName(Size)
{//Size 是数组的长度
   this.length=Size;
   for(var i=0; i<Size;i++)
         this[i]=0;
   return this;
}
```

其中，arrayName 是数组对象名；Size 是数组的大小，通过 for 循环对一个当前对象的数组进行定义；最后返回这个数组。从定义可以看出，实际上定义了这样一个对象，它没有单独的属性名，通过 this[i]对它的属性赋值。

② 创建数组实例。一个数组对象定义完成以后，还不能马上使用，必须使用 new 操作符为该数组创建一个数组实例。例如：

```
MyArray=new arrayName(10);
```

并为各元素赋初值：

```
MyArray[0]= 1；
MyArray[1]= 2；
...
MyArray[9]= 10；
```

一旦给数组元素赋予了初值，数组中就有了真正意义的数据，便可以在程序中引用。

了解这种数组的实现方式，可帮助我们理解数组对象的本质。但这种实现方式与直接使用 Array 对象相比要复杂一些，所以在实际应用中，还是使用 Array 对象来实现数组更方便。

2．String 对象

前面的例子中已经多次使用了字符串，在 JavaScript 中每个字符串都是对象。

（1）创建 String 对象实例。创建 String 对象实例的语法是：

```
[var] String 对象实例名=new String(string);
```

或

```
var String 对象实例名=字符串值；
```

例如：

```
str1=new String("This is a sample. ");
str2="This is a sample. ";
```

以上两种格式定义效果完全相同，我们通常更习惯于用后者，它是一种“隐式”创建对象实例方式（即不使用 new 保留字）。

（2）String 对象的属性。String 对象的属性只有一个：length（长度），其值是字符串包含的字符个数。

例如，对上面定义的字符串 str2，str2.length 的值为 17。

（3）String 对象的方法。String 对象的方法较多，共有 19 个，下面讨论常用的 8 类。

① charAt(position)。返回 String 对象实例中位于 position 位置上的字符，其中 position 为正整数或 0。注意字符串中字符位置从 0 开始计算。

② indexOf(str)、indexOf(str,start-position)。字符串查找，str 是待查找的字符串。在 String 对象实例中查找 str，若给出 start-position，则从 start-position 位置开始查找，否则从 0 开始查找；若找到，返回 str 在 String 对象实例中的起始位置，否则返回-1。例如：

```
var str1="This is a sample. ";        //str1.length 值为 17
var str2="sample";
found=str1.indexOf(str2);             //found 的值为 10
```

③ lastIndexOf(str)。该方法与 indexOf()类似，区别在于它是从右往左查找。

④ substring(position)、substring(position1,position2)。返回 String 对象的子串。如果只给出 position，返回从 position 开始至字符串结束的子串；如果给出 position1 和 position2，则返回从二者中较小值处开始至较大值处结束的子串。例如对上面定义的 str1，str1.substring(2,6)和 str1.substr(6,2)都返回"is i"。

⑤ toLowerCase()、toUpperCase()。分别将 String 对象实例中的所有字符改变为小写或大写。

⑥ 有关字符显示的控制方法。big()为大字体显示, italics()为斜体字显示, bold()为粗体字显示, blink()为字符闪烁显示，small()为字符用小体字显示，fixed()为固定高亮字显示，fontsize(size)为控制字体大小等。

⑦ 锚点方法 anchor()和超链接方法 link()。锚点方法 anchor()返回一个字符串，该字符串是网页中的一个锚点名。使用 anchor()与用 HTML 中的标记<A Name="">的作用相同。该方法的语法格式为：

```
string.anchor(anchorName)
```

例如：

```
var astr = "开始";
var aname = astr.anchor("start");
document.write(aname);
```

上述语句将在网页中创建一个名为 start 的锚点，而该锚点处显示文字“开始”。这几条语句与下面的 HTML 标记作用相同：

```
<a name="start">开始</a>
```

超链接方法 link()返回一个字符串，该字符串在网页中构造一个超链接，其语法格式为：

```
string.link(href)
```

其中，href 是超链接的 URL。

例如：

```
var hstr="去新浪"；
```

```
var hLoc=hstr.link("http://www.sina.com.cn");
```

上述两条语句等价于在 HTML 文件中使用以下标记：

```
<a href="http://www.sina.com.cn">去新浪</a>
```

⑧ fontcolor(color)、fontsize()。字号方法 fontsize()的使用与 fontcolor(color)基本相同。字体颜色方法 fontcolor(color)返回一个字符串，此字符串可改变网页中的文字颜色。语法格式为：

```
str.fontcolor(FontColor)
```

其中，FontColor 是颜色值，可以是一个英文单词，或一个十六进制数值，详见第 3 章。

例如：

```
str="红色文字";
strColor=str.fontcolor("red");
document.write(strColor);
```

上述语句相当于在 HTML 文件中使用以下标记：

```
<font color=red>红色文字</font>
```

字体大小方法 fontsize()的使用与 fontcolor(color)基本相同。

3．Math 对象

Math 对象封装了常用的数学常数和运算，包括三角函数、对数函数、指数函数等。Math 对象与其他对象不同，它本身就是一个实例，是由系统创建的，称为“静态对象”，不能用 new 创建 Math 对象实例。

（1）Math 对象的属性。Math 对象的属性定义了一些常用的数学常数，它们是只读的。这些属性列于表 5-5 中。

表 5-5　Math 对象属性表

属　性　名	含　　义
E	常数 e，自然对数的底，近似值为 2.718
LN2	2 的自然对数，近似值为 0.693
LN10	10 的自然对数，近似值为 2.302
LOG2E	以 2 为底，e 的对数，即 $\log_2 e$，近似值为 1.442
LOG10E	以 10 为底，e 的对数，即 $\log_{10} e$，近似值为 0.434
PI	圆周率，近似值为 3.142
SQRT1_2	0.5 的平方根，近似值为 0.707
SQRT2	2 的平方根，近似值为 1.414

例如，引用自然对数的底 e，格式为 Math.E。

（2）Math 对象的方法。Math 对象的方法包括三角函数、对数和指数函数及舍入函数等。表 5-6 列出了常用的一些方法。例如，要使用正弦三角函数，格式为：Math.sin(3.2)。

表 5-6　Math 对象常用方法

方　　法	含　　义
sin(val)	返回 val 的正弦值，val 的单位是 rad（弧度）
cos(val)	返回 val 的余弦值，val 的单位是 rad（弧度）
tan(val)	返回 val 的正切值，val 的单位是 rad（弧度）
asin(val)	返回 val 的反正弦值，val 的单位是 rad（弧度）
exp(val)	返回 e 的 val 次方
log(val)	返回 val 的自然对数
pow(bv,ev)	返回 bv 的 ev 次方
sqrt(val)	返回 val 的平方根

续表

方　法	含　义
abs(val)	返回 val 的绝对值
ceil(val)	返回大于或等于 val 的最小整数值
floor(val)	返回小于或等于 val 的最小整数值
round(val)	返回 val 四舍五入得到的整数值
random()	返回 0～1 之间的随机数
max(val1,val2)	返回 val1 和 val2 之间的大者
min(val1,val2)	返回 val1 和 val2 之间的小者

4．Date 对象

Date 对象封装了有关日期和时间的操作，它有大量设置、获得和处理日期和时间的方法，但没有任何属性。

（1）创建 Date 对象实例。创建 Date 对象实例的语法是：

```
[var] Date 对象名=new Date([parameters]);
```

参数可以是以下的任一种形式。

无参数　　获得当前日期和时间。

形如“月 日, 年 时：分：秒”的参数　　创建指定日期和时间的实例。

形如“年、月、日、时、分、秒”的整数值参数　　创建指定日期和时间的实例（省略时、分、秒，其值将设为 0）。

例如：

```
var today=new Date();
birthday=new Date("September 10,1990 5:50:20");
birthday=new Date(90,9,20);
birthday=new Date(90,9,20,5,50,20);
```

（2）Date 对象的方法。Date 对象的方法可分为以下 4 类。

① get 方法组，在 Date 对象中获取日期和时间值。主要包括以下 9 种。

getYear()：返回对象实例的年份值。如果年份在 1900 年后，则返回后两位，如 1998 将返回 98；如果年份在 100～1900 之间，则返回完全值。

getMonth()：返回对象实例的月份值，其值在 0～11 之间。

getDate()：返回对象实例日期中的天，其值在 1～31 之间。

getDay()：返回对象实例日期是星期几，其值在 0～6 之间，0 代表星期日。

getHours()：返回对象实例时间的小时值，其值在 0～23 之间。

getMinutes()：返回对象实例时间的分钟值，其值在 0～59 之间。

getSeconds()：返回对象实例时间的秒值，其值在 0～59 之间。

getTime()：返回一个整数值，该值等于从 1970 年 1 月 1 日 00:00:00 到该对象实例存储的时间所经过的毫秒数。

getTimezoneOffset()：返回当地时区与 GMT 标准时的差别，单位是 min。（GMT 时间是基于格林尼治时间的标准时间，也称 UTC 时间）。

② set 方法组，设置 Date 对象中的日期和时间值，包括 setYear(year)、setMonth(month)、setDate(date)、setHours(hours)、setMinutes(minutes)、setSeconds(senconds)和 setTime(time)，含义与 get 方法组相同。

③ to 方法组，从 Date 对象中返回日期和时间的字符串值，包括 toGMTString()、toLocalString()

和 toString()。

④ parse()和 UTC()方法，用于分析 Date 字符串。这两个方法的用法比较特殊，它们是由 Date 对象本身（也称系统实例）使用的，通常称这样的方法为静态成员方法。parse()方法的语法为：

```
Date.parse(DateString);
```

它将字符串参数表示的日期转换为一个整数值，该值等于从 1970 年 1 月 1 日 00:00:00 计算起的毫秒数。

UTC 方法的语法为：

```
Date.UTC(year,month,date,hour,minute,second);
```

它将数值参数表示的日期转换为一个整数值，该值等于从 1970 年 1 月 1 日 00:00:00 起计算的毫秒数。

例如：

```
date1=new Date();
date2=new Date();
date1.setTime(Date.parse("10 Jan,2000 20:10:10"));
date2.setTime(Date.UTC(2000,9,1,20,10,10));
```

parse()方法的字符串参数可有多种形式，例如：

```
"10 Jan,2000 20:10:10"
"Jan 10,2000"
"10 Jan 2000"
"1/10/2000"
"10 Jan,2000 20:10:10 GMT" //当字符串末尾有 GMT 时，计算机的时区设置将起作用。
```

parse()与 UTC()的区别在于参数不同，前者参数为字符串，后者参数为整数。它们可以用于日期的比较。

【例 5-10】一个有关 Date 对象的应用例子。该 HTML 文件在浏览器窗口显示一个不断刷新的数字时钟。程序的运行结果如图 5-9 所示。

```
<html><head><title>数字钟</title>
  <style>
    form { font-size:22px; }
    input { font-size:24px;
            color:red;
            width:180;height:40;}
</style>
<script language="JavaScript">
function aClock(){
      var now=new Date();
      var hour=now.getHours();
      var min=now.getMinutes();
      var sec=now.getSeconds();
      var timeStr=" "+hour;
      timeStr+=((min<10)?":0":":")+min;
      timeStr+=((sec<10)?":0":":")+sec;
      timeStr+=(hour>=12)?" P.M.":" A.M.";
      document.clock_form.clock_text.value=timeStr;
      clockId=setTimeout("aClock()",1000);
    }
</script></head>
<body onLoad="aClock()">
```

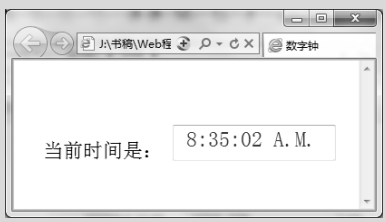


图 5-9　不断刷新的数字时钟

```
<br><br><br>
<form name="clock_form">
      当前时间是：
    <input type="text" name="clock_text" value="">
</form></body></html>
```

例 5-10 在头部定义了函数 aClock()，该函数应用 Date 对象的多个方法。函数 aClock()中最关键的是语句 clockId=setTimeout("aClock()",1000)；其中 setTimeout()是内建函数，setTimeout ("aClock()",1000)表示每隔 1 000 ms 调用 1 次 aClock()。所以本例设计的数字时钟每隔 1 秒刷新 1 次。本例 body 在页面载入时用 onLoad 事件处理将 aClock()调入执行。

5．Number 对象

Number 对象给出了系统最大值、最小值及非数字常量的定义，这些常量的定义列于表 5-7 中。

表 5-7　Number 对象属性表

属　　性	含　　义
MAX_VALUE	数值型最大值，值为 1.7976931348623517e+308
MIN_VALUE	数值型最小值，值为 5e-324
NaN	非合法数字值
POSITIVE_INFINITY	正无穷大
NEGATIVE_INFINITY	负无穷大

Number 对象与 Math 对象一样，也是静态对象，因此引用表 5-7 中的 Number 对象属性的格式也与 Math 对象相似。例如：

```
Max_val=Number. MAX_VALUE;
```

6．Boolean 对象

Boolean 对象的作用是将非布尔量转换为布尔量。以下语法可创建 Boolean 对象实例：

```
[var] BoolVal=new Boolean([参数])；
```

其中，参数可以为空。当参数为空，或参数为 0、null、false、空字符串时，所创建的对象实例为 false；其他情况下，所创建的对象实例为 true。例如：

```
var BoolVal1=new Boolean();              //BoolVal1 值为 false
var BoolVal2=new Boolean("");            //BoolVal2 值为 false
var BoolVal3=new Boolean(8);             //BoolVal3 值为 true
var BoolVal4=new Boolean("This");        //BoolVal4 值为 true
```

7．Function 对象

前面已经讨论并多次应用了函数的定义，Function 对象提供了另一种定义和使用函数的方法。利用 Fonction 对象定义函数对象实例的语法为：

```
var FuncName=new Function([arg1],[arg2],...,FuncString);
```

其中，FuncName 是函数名，arg1，arg2，…，是函数的形式参数，它们可以没有；FuncString 是字符串形式的函数体。这样定义的函数实例，可以像普通函数一样调用。例如：

```
var setColor=new Function(" document.color='darkgreen'");
```

以后就可以调用它。例如：

```
if (MustSetColor) then { serColor(); }
```

利用 Function 对象定义和调用函数与前面讨论的函数定义和使用方法相比，后者在执行速度上更快一些，所以以使用后一种方法为主。

8．预定义函数

预定义函数不属于任何对象，不必通过对象来引用它们。

（1）eval()函数。其语法为：

```
eval(string);
```

其中，string 是一个字符串，它的内容应是一个合法表达式。eval()函数将表达式求值，返回该值。例如：

```
var sum=eval("2+3*4");      //sum 的值为 14
var a=2;
var val=eval("5+3*a");      //val 的值为 11
```

（2）isNaN()函数。其语法为：

```
isNaN(testValue)；
```

其中，testValue 是被测试的表达式，它可以是任意类型的表达式。isNaN()测试表达式的值是否为 NaN，若是，isNaN()返回 true；否则返回 false。注意，有些平台不支持 NaN 常量，则此函数无效。

（3）parseInt()和 parseFloat()函数。

parseInt()函数的语法为：

```
parseInt(str[,radix])；
```

其中，str 是一个字符串。可选参数 radix 是整数，若给出，则表示基数；若未给出，则表示基数为 10。parseInt()函数先对字符串形式的表达式求值，若求出的值是整数，则转换为相应基数的数值。若不能求出整数值，则返回 NaN 或 0。

parseFloat()函数的语法为：

```
parseFloat(str)；
```

parseFloat()函数的使用与 parseInt()类似，其所求的值为浮点数。例如：

```
floatVal=parseFloat("1e28");
if (isNaN(floatVal)){
        document.write("Not float");
}
else {
        document.write("Is float");
}
```

5.2　浏览器对象模型及应用

浏览器对象模型（Browser Object Model，BOM）将网页处理为对象的集合，网页元素都可以是对象，具有属性、方法和事件，通过脚本语言就可以操作网页元素。浏览器对象模型提供了用户与浏览器之间的交互的对象及操作的接口。这些对象不仅有用于操作页面内容的，还有用来读取客户端信息及对客户端系统进行操作，例如代表屏幕的 screen 对象，代表浏览器的 navigator 对象等。现代浏览器基本上都实现了 JavaScript 交互性方面的相同方法和属性，因此这些方法和属性常被认为是 BOM 的方法和属性。

浏览器对象之间并不是独立存在的，浏览器窗口与网页之间、网页与网页各组成部分之间都具有一定的从属关系。如 window 对象是 document 对象的父对象等。浏览器对象模型就是用于描述这种对象与对象之间层次关系的模型，该对象模型提供了独立于内容，可以与浏览器窗口进行互动的对象结构。

5.2.1 浏览器对象模型

浏览器对象模型是按照层次组织的，从而形成树形结构，称为 Navigator 对象树，它对页面设计非常重要。Navigator 对象层次结构如图 5-10 所示。

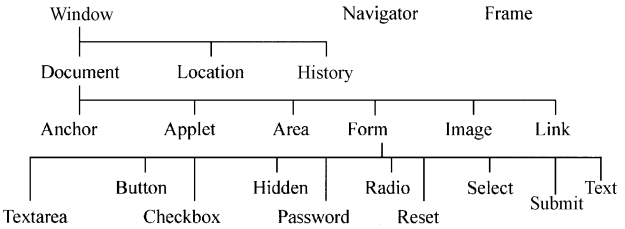


图 5-10　Navigator 对象层次结构

该结构中有三个顶层对象 Window、Navigator、Frame，常用对象的含义如下。

（1）Navigator 对象：封装了浏览器名称、版本、客户端支持的 mime 类型等环境信息。

（2）Window 对象：封装了有关窗口的属性和窗口操作。

（3）Frame 对象：在浏览器中使用多个窗口时用到该对象，它与 Window 对象相似，对应子窗口。

（4）Location 对象：包含基于当前 URL 的信息。

（5）History 对象：包含浏览器的浏览历史信息。

（6）Document 对象：最重要的对象之一，代表当前 HTML 文件。

（7）Form 对象：包含表单的属性和操作。

（8）Anchor 对象：包含页面中锚点的信息。

（9）Button、Password、Checkbox 等对象：是 Form 的下层对象，对应 Form 中相应元素。

Navigator 对象层次结构中列出的对象并非在每个 HTML 文件中都出现，例如有些页面并不使用 Form，因此它就没有 Form 对象。但有几个对象是每个 HTML 文件都有的，它们是：Window、Navigator、Document、Location 和 History。

5.2.2 Navigator 对象

Navigator 对象包含正在使用的浏览器版本信息，包括 appName、appVersion、AppCodeName、userAgent、mimeType、plugins 属性和 javaEnabled()、taintEnabled()方法。Navigator 对象的主要用途是判别客户浏览器的类别，以便针对不同浏览器的特性而设计不同的显示。Navigator 对象常用的属性和方法的含义见表 5-8。

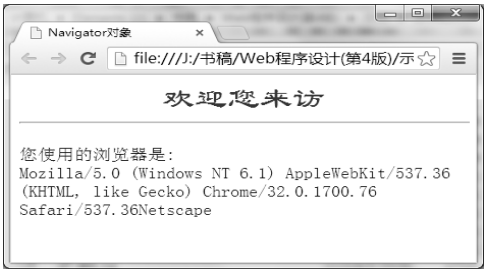
表 5-8　Navigator 对象常用属性和方法表

属性或方法名	含　义
appName	以字符串形式表示浏览器名称
appVersion	以字符串形式表示浏览器版本信息，包括浏览器的版本号、操作系统名称等
appCodeName	以字符串形式表示浏览器代码名字，通常值为 Mozilla
userAgent	以字符串表示完整的浏览器版本信息，包括 appName、appVersion、appCodeName 信息
mimeType	在浏览器中可以使用的 mime 类型
plugins	在浏览器中可以使用的插件①
javaEnabled()	返回逻辑值，表示客户浏览器可否使用 Java

注：①mimeTypes 和 plugins 是两个数组，其元素分别是 MimeType 对象和 Plugin 对象。

【例 5-11】根据浏览器的类型显示不同的页面，在 Chrome 和 IE 中的结果如图 5-11 所示。

```
<html><head><title>Navigator 对象</title></head>
<body><center>
<font face="隶书" color=red size=6>欢迎您来访</font></center>
<script language="JavaScript">
        if (navigator.appName=="Netscape")
              document.write ('<hr width=100%>');
        else {   document.write('<font face="隶书" color=darkgreen size=4>');
                 document.write('<marquee border="0">您好！欢迎您来到我的主页
                   </marquee></font>');
              }
        document.write("<br><font size=4 color=blue>您使用的浏览器是:<br>");
        document.write(navigator.userAgent);
        document.write(navigator.appName);
        document.write("</font>");
</script></body></html>
```



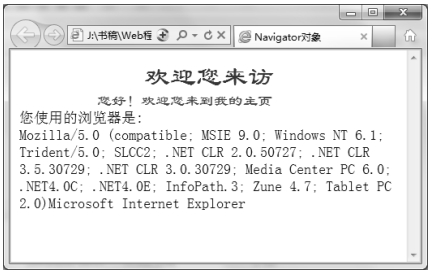


图 5-11　Navigator 对象示例

5.2.3　Window 对象

Window 对象描述浏览器窗口特征，它是 Document、Location、History 对象的父对象。另外，Window 对象还可认为是其他任何对象的假定父对象，如语句“ alert("世界，你好！"); ”相当于语句“ window.alert("世界，你好！"); ”。

Window 对象的属性有 parent、self、top、window、status、defaultStatus、frames 等，方法有 alert()、open()、close()、confirm()、prompt()、focus()、blur()、setTimeout()、clearTimeout()等，以下分类讨论。

（1）与窗口有关的属性

包括 parent、self、top、window，这 4 个属性是特殊的，严格来说，它们并不能算做 Window 对象的属性，而是当前浏览器环境所涉及的 Window 对象的实例。因此它们的引用与一般对象属性不同：在它们的名称之前不能加对象名，如 self.status，而不是 window.self.status。

window 和 self 代表当前窗口；parent 代表当前窗口或帧（frame）的父窗口，主要在使用帧的页面中使用；top 是主窗口，是所有下级窗口的父窗口。

（2）与浏览器状态栏有关的属性

包括 status、defaultStatus, 其值都为字符串。status 是浏览器当前状态栏显示的内容, defaultStatus 是浏览器状态栏显示的默认值。利用这两个属性可以设置和改变浏览器状态栏显示的内容。例如：

```
<a href="http://www.163.com" onMouseOver="status='访问网易'; return true">网易<br>
//当鼠标位于该超链接位置时，状态栏将显示字符串“ 访问网易 ”。
```

（3）与对话框有关的方法

包括 alert()、confirm()和 prompt()三个方法，它们分别产生三个标准对话框。它们的语法分别为：

① alert(字符串); //参数字符串为显示于对话框中的内容，无返回值；
② confirm(字符串); //参数字符串为显示于对话框中的内容。若用户单击“确定”按钮，返回值为 true，否则返回值为 false；
③ prompt(字符串 1，字符串 2); //参数字符串 1 为显示于对话框中的内容，参数字符串输入的默认内容；如用户单击对话框的“确定”按钮，则返回用户在输入框中输入的字符串；如用户单击对话框的“取消”按钮，则返回 null。

例如：

```
alert("你好！");
confirm("你确定要继续吗？");
prompt("请输入您的姓名：","*******");
```

这三条语句所产生的对话框分别如图 5-12（a）、（b）、（c）所示。注意不同的浏览器外观显示会有所不同。

（a）alert()生成的对话框

（b）confirm()生成的对话框

（c）prompt()生成的对话框

图 5-12　与对话框有关的方法示例

（4）与窗口生成与撤销有关的方法

包括 open()、close()方法。open()方法生成一个新窗口，语法为：

```
open("URL", "WindowName" [,"Window Features"]);
```

其中，参数 URL 是在新生成的窗口中载入的页面；WindowName 是新窗口的名字；Window Feature 是可选参数，该参数是一个字符串，表示新窗口的外观特征，可以指定多个特征，各特征值之间以逗号“，”相隔，特征值格式为：特征名=值，如 width=300，表示新窗口的宽度为 300 个像素点。该参数省略时，按默认特征生成新窗口。各特征值列于表 5-9 中。open()返回指向新窗口的指针。

表 5-9　窗口特征值表

特　征　名	取　　值	含　　义
width	长度值	窗口的宽度
height	长度值	窗口的高度
toolbar	0（无）|1（有）或 No（无）|yes（有）下同	是否显示标准工具栏。默认值为 0
location	0|1 或 no|yes	是否显示定位栏。默认值为 0
status	0|1 或 no|yes	是否显示状态栏。默认值为 0
menubar	0|1 或 no|yes	是否显示菜单栏。默认值为 0
srcollbars	0|1 或 no|yes	是否按需要显示滚动条。默认值为 0
resizable	0|1 或 no|yes	是否允许用户改变窗口大小。默认值为 1

例如，语句：

```
nw=open("a.htm","nw","width=100,height=80,toolbar=1,resizable=0");
```

将创建一个名为 nw 的新窗口，其中载入页面 a.htm，该窗口宽为 100，高为 80，用户不可改变显示工具栏的大小。注意各特征值与先后顺序无关。

方法 close()用于关闭一个窗口。例如：

```
nw.close();
```

（5）与窗口焦点有关的方法

包括 focus()、blur()方法。支持多窗口操作的操作系统在任何时刻都只有一个窗口处于“ 激活 ”状态，可以接收用户的输入，这样的窗口我们就称它获得了焦点，否则称它失去了焦点。方法 focus()、blur()使窗口分别获得和失去焦点，例如：

```
nw.focus();
nw.blur();
```

（6）与“ 超时 ”有关的方法

包括 setTimeout()、clearTimeout()方法。方法 setTimeout()意为“ 设置超时 ”，其语法为：

```
setTimeout("expression",time);
```

其中, 参数 expression 是一个表达式, 通常为一个函数 ;time 是整数, 单位是 ms。执行 setTimeout()的结果是每隔 time ms 将重新对 expression 求值 1 次。setTimeout()返回一个标志，指示这个“ 超时 ”设置。

clearTimeout()方法的作用是清除指定的超时设置。语法为：

```
clearTimeout(timeId);    //参数 timeID 是由 setTimeout 返回的标志
```

（7）其他方法

① 属性 opener，是一个窗口名，该窗口是由 open()打开的最新窗口。

② 属性 frames，是一个数组，数组的各成员是窗口内的各帧。

③ 方法 scroll(x,y)，使窗口滚动到 x、y 处。

另外, HTML 文件被载入和退出窗口, 分别会触发 Load()和 Unload()事件, 即在 HTML 的 body 元素中可以设置 onLoad 和 onUnload 事件处理代码。

Window 对象内容丰富，在页面设计时可以充分利用该对象提供的属性和方法。以下举一个例子总结 Window 对象的主要属性和方法。

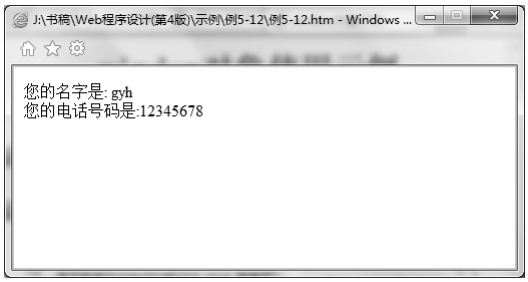
【例 5-12】在浏览器中显示一个如图 5-13（a）所示的初始用户输入界面，接收用户输入的姓名和电话号码，用户单击“ 输入完成 ”按钮后，将弹出如图 5-13（b）所示的对话框，要求用户再次确认，若单击“ 确认 ”按钮，则生成一个如图 5-13（c）所示的新窗口，显示用户输入的姓名和电话号码。若单击图 5-13（b）中的“ 取消 ”按钮，则弹出如图 5-13（d）所示的警告框。

```
<html><head><title>window 对象示例</title>
<script language= "JavaScript">
function confSubmit(){
if ((ok=confirm("您确定输入正确吗？"))==true){
var nw=open("a.htm","nwin","width=500,height=200,toolbar=1");
nw.focus();
nw.document.write("您的名字是:"+parent.document.input_form.nm.value);
nw.document.write("<br>");
nw.document.write("您的电话号码是:" + parent.document.input_form.phone.value);
}
else
   alert("请您重新输入！");
}
</script></head>
<body>
```

```
<script>defaultStatus="这是一个 window 对象使用示例。";</script>
<center><h1>window 对象使用示例一</h1></center><hr><br>
<form name="input_form"><h2>
请输入您的姓名：<input type=text name="nm" size=12 value=" "><br><br>
请输入您的电话号码：<input type=text name="phone" size=12><br><br>
  <input type=submit value="确定" onClick="confSubmit()"></h2>
</form></body></html>
```

（a）初始用户输入界面

（b）单击“输入完成”按钮后的对话框



（c）单击“确定”按钮生成的新窗口

（d）单击“取消”按钮弹出的对话框

图 5-13　Window 对象使用示例

函数 confSubmit()在 form 的“submit”按钮被单击后调用，该函数首先弹出一个 confirm 类型的对话框，然后根据用户的选择进行不同的操作，要注意的是用户单击“确定”按钮后的操作：利用 Window 对象的 open()方法生成新窗口“nwin”，新窗口指针为 nw，在后面的对新窗口所对应的 HTML 文件的写入操作时，引用窗口对象是通过窗口指针而非窗口名；向新窗口对应 HTML 文件写入的内容是用户在原窗口输入的值，故引用这些值时须指明值所属的对象，用 parent 表示引用的是新窗口的父窗口对象。

5.2.4　Document 对象

前面的例子已经多次引用了 Document 对象的 write()方法，该方法可以向 Document 对象所对应的 HTML 文件写入内容。一个 HTML 文件的页面对应一个 Document 对象，通过 Document 对象的属性和方法，可以创建 HTML 文件，所以它是浏览器对象中最有用的对象之一。

1．Document 对象的属性

Document 对象的属性较多，包括数值属性和对象数组属性。

（1）数值属性

数值属性是指 Document 对象的数值变量形式的属性，该属性本身不是任何对象或数组。

Document 对象的数值属性大部分与 HTML 标记相对应，主要用于设置和改变页面的背景、文本、超链接颜色等显示特性。常用的数值属性列于表 5-10 中。

表 5-10　Document 对象的数值属性

属 性 名	取　值	含　义
alinkColor	颜色值	被激活的超链接文本颜色，即鼠标单击超链接时超链接文本的颜色
bgColor	颜色值	页面背景颜色
fgColor	颜色值	页面前景颜色，即页面文字的颜色
lastModified	日期字符串	HTML 文件最后被修改的日期，是只读属性
linkColor	颜色值	未被访问的超链接的文本颜色
referrer	URL 字符串	用户先前访问的 URL
title	字符串	HTML 文件的标题，对应<title>标记
URL	URL 字符串	本 HTML 文件完整的 URL
vlinkColor	颜色值	已被访问过的超链接的文本颜色

【例 5-13】通过 JavaScript 设置页面的颜色和文字等属性。页面显示效果如图 5-14 所示。

```
<html><head><title>通过 Document 对象设置页面属性</title></head>
<body><script language= "JavaScript">
document.bgColor="#DDEEFF";
document.fgColor="darkred";
document.linkColor="#0088FF";
document.alinkColor="#0088FF";
document.vlinkColor="#0088FF";
document.write("<h1>通过 Document 对象设置页面属性示例</h1>");
document.write("<hr>");
document.write("<a href='a.htm'>去页面 A</a>");
document.write("<br><br>本 HTML 文件名是："+document.URL);
document.write("<br><br>本 HTML 文件最后被修改的时间是："+document.lastModified);
</script></body></html>
```

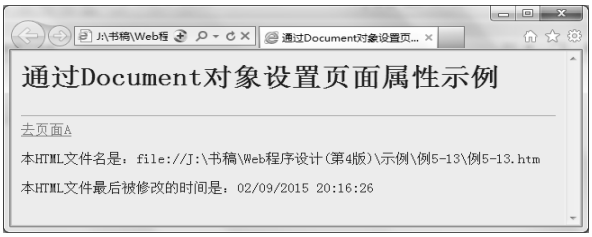


图 5-14　Document 对象的数值属性示例

（2）对象数组属性

Document 的对象数组属性包括 anchors、applets、forms、images、links 等，分别反映一个 HTML 文件中的锚点、Java Applet、表单、图像、超链接信息。

① anchor 对象和 anchors 数组

HTML 定义锚点（即超链接位置）的语法为：

```
<a [href=location 或 URL] name="anchorName"
    [target=windowName]>anchorText</a>
```

其中，以<a>标记的 name 属性来命名锚点，一个命名锚点对应一个 anchor 对象。anchors 数组是 HTML 文件中 anchor 对象的序列。anchor 对象没有属性，所以 anchors 数组是只读的。anchors 数组有一个属性 length，表示该数组的长度，即 HTML 文件中所命名锚点的数目。

【例 5-14】设计如图 5-15 所示的框架网页。窗口划分为左、右两个框架，文件 anchor1.htm 显示于左框架中，它包含一系列按钮，每个按钮对应右框架中显示的页面锚点；文件 anchor2.htm 显示于右框架中，它定义了被命名为“0”，“1”，“2”和“3”的 4 个锚点。当单击左框架中的某按

钮时，右框架中显示被定位到的指定目标。主文件名为 anch.htm。

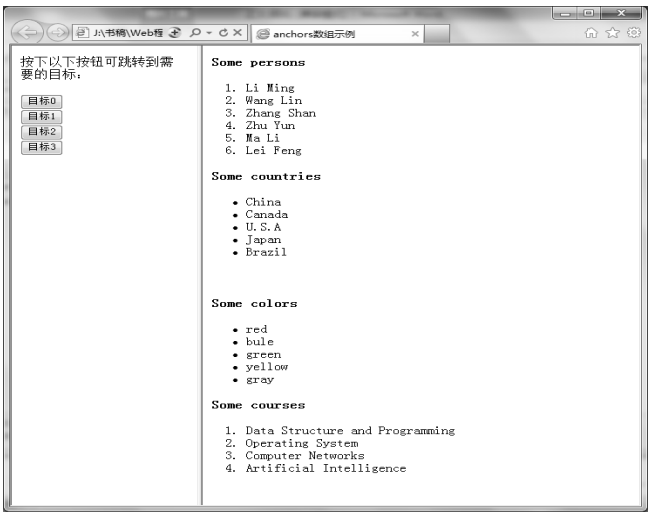


图 5-15　Document 对象的 anchors 数组的使用示例

文件 anch.htm 的内容：

```
<html><head><title>anchors 数组示例</title></head>
<frameset cols="30%,*">
<frame src="anchor1.htm">
<frame src="anchor2.htm" name="anchors2">
</frameset></html>
```

文件 anchor1.htm 的内容：

```
<html><head><title>anchors:frame 1</title></head>
<body><script language= "JavaScript">
function linkToAnchor(num){
   if (parent.anchors2.document.anchors.length>=num)
         parent.anchors2.location.hash=num
   else
         alert("目标不存在！");
}
</script>
单击以下按钮可跳转到需要的目标：<br><br>
<form><input type="button" value="目标 0"
       name="anch0" onclick="linkToAnchor(0)"><br>
<input type="button" value="目标 1" name="anch1" onclick="linkToAnchor(1)"><br>
<input type="button" value="目标 2" name="anch2" onclick="linkToAnchor(2)"><br>
<input type="button" value="目标 3" name="anch3" onclick="linkToAnchor(3)">
</form></body></html>
```

文件 anchor2.htm 的内容：

```
<html><head><title>anchors:frame 2</title></head>
<body><p><a name="0"><b>Some persons</b></a>
<ol><li>Li Ming<li>Wang Lin<li>Zhang Shan<li>Zhu Yun<li>Ma Li<li>Lei Feng
</ol></p>
<p><a name="1"><b>Some countries</b></a>
<ul><li>China<li>Canada<li>U.S.A<li>Japan<li>Brazil</ul></p>
<p><a name="2"><b>Some colors</b></a>
<ul><li>red<li>bule<li>green<li>yellow<li>gray</ul></p>
<p><a name="3"><b>Some courses</b></a><ol><li>Data Structure and Programming
```

```
<li>Operating System<li>Computer Networks<li>Artificial Intelligence</ol></p>
</body></html>
```

在例 5-14 中需注意的是文件 anchor1.htm 中函数 linkToAnchor(num)中的语句：

```
if (parent.anchors2.document.anchors.length>=num)
    parent.anchors2.location.hash=num
```

因为通过 anchors 无法直接得到锚点名，所以这里通过 location 对象的 hash 属性来设置锚点。location 对象可决定浏览器窗口的 URL，通过设置 location.hash 即可决定在 HTML 文件中的锚点。

② image 对象和 images 数组

HTML 文件中的一个<img>标记对应一个 image 对象, 即 image 对象将页面中的图像信息封装了起来，image 对象的属性与<img>标记的属性相对应。一个较完整的<img>定义如下：

```
<img name="img1" src="bk.gif" hspace="30" vspace="30" width="100" height="80"
     border="2" lowsrc="bk1.gif">
```

与<img>标记的定义相对应，image 对象的属性如下。

name　　//img 标记的名字，在 JavaScript 程序中可通过该名字引用对应的 image 对象。
src　　//图像文件的 URL。
width　　//图像的宽度。
height　　//图像的高度。
border　　//图像边框的宽度。
hspace　　//图像与左边或右边文字的空白大小。
vspace　　//图像与上边或下边文字的空白大小。
lowsrc　　//在 src 所指出的图像文件装载完之前显示的图像。

此外，image 对象还有 complete 属性，它是一个布尔值，表示图像文件是否装载完成。

一个 HTML 文件中各个<img>标记所对应的 image 对象，按照它们在文件中出现的先后顺序形成数组 images。通过 images 数组，可以按照需要动态地改变某些图像文件。例如，可以根据当前日期决定显示的图像，根据鼠标的位置决定超链接图像的内容以及实现动画显示等，这些是只使用 HTML 标记所不能完成的。

【例 5-15】实现一个简单的小动画，交替显示三幅图像。

```
<html><head><script language="JavaScript">
var ImageNum=1;
function Begin(){
   document.MyImage.src=ImageArray[ImageNum].src;
   ImageNum++;
   if (ImageNum>3)
         ImageNum=1;
   }
</script></head>
<body>
<img name="MyImage" src="images/alvbull1.gif" onLoad= "setTimeout('Begin()',100)">
<script language="JavaScript">
var ImageArray=new Array();
for (i=1;i<=3;i++)
{    ImageArray[i]=new Image();
     ImageArray[i].src="images/alvbull"+i+".gif";
}
</script></body></html>
```

例 5-15 事先准备了三个图像文件，分别为 alvabull1.gif、alvabull2.gif、alvabull3.gif，程序每隔 0.1 秒更换一次所显示的图像文件，从而产生动画效果。

（3）链接对象和链接数组

链接数组提供 HTML 文件中的超链接，通过它可以得到超链接的信息并加以控制。链接数组的每个对象都是一个链接对象（Link）或 Area 对象。

Link 对象存储 URL 的信息，一个完整的 URL 包括协议、主机名或 IP 地址、协议端口号、路径名、hash 数。例如下面的 URL：

```
http://www.njim.edu.cn:2000/java/index.html#follow-up
```

对应的各个部分的 Link 对象的属性如下。

```
hash        //对应 hash 数，即锚点名，如#follow-up。
host        //主机名或主机 IP 地址，如 www.njim.edu.cn。
hostname    //主机和端口的组合，如 www.njim.edu.cn:2000。
href        //代表整个 URL。
pathname    //路径，如/java/index.html。
port        //服务器端口号，如 2000。
protocol    //代表协议，如 http。
search      //查询信息，上例中没有对应部分。
```

查询数据前加一个问号，这些数据包含在 URL 的最后一项，格式为：

```
？name=value
```

Area（位图映射机制）通过位图的不同位置来实现不同的链接。Area 对象在 JavaScript 中被作为特殊的 link，故被归到 links 数组中。Area 对象的属性与 Link 对象的属性完全相同。

【例 5-16】links 数组用法示例。设计如图 5-16 所示的页面，利用按钮改变超链接的对象。

```
<html>
<head><title>links 数组使用示例</title></head>
<body><br><br><br><br>    选择以下按钮决定
<a href="javascript:alert('请按按钮选择下一步访问的站点！')">访问哪个站点。</a>
<form><br>        
<input type="button" value="新　浪　网"
    onClick="document.links[0].href='http://www.sina.com.cn'"><br>  
<input type="button" value="搜　　狐"
    onClick="document.links[0].href='http://www.sohu.com'"><br>  
<input type="button" value="中央电视台"
    onClick="document.links[0].href='http://www.cctv.com.cn'">
</form>
</body>
</html>
```

图 5-16　links 数组的用法示例

例 5-16 中，单击所要访问站点的按钮（如“ 中央电视台 ”），再单击“ 访问哪个站点 ”超链接，就将链接到所选择的站点。例 5-16 的 HTML 文件中只包含一个超链接，该超链接的值是根据用户单击的按钮决定的。本例的<a>标记的 href 属性值被赋为“ javascript:alert('请按按钮选择下一步访

问的站点！')”，这里的“ javascript: ”表示使用 JavaScript 协议，在它之后应跟 JavaScript 语句，这是 JavaScript 的另一种使用方式。

2．Document 对象的方法

Document 对象的方法主要有 write()、writeln()、open()、close()、clear()。

（1）write()和 writeln()方法

write()方法已经用过许多次了，用于输出内容到 HTML 文件中。其语法是：

```
write(String1,String2,...);
```

write()的参数可以是任意多个字符串，但至少有一个。当参数不是字符串时，将被自动转换为字符串。例如：

```
document.write("欢迎访问本主页！");
document.write("您是第"+i+"个访问本主页的贵宾");
document.write("您是第",i,"个访问本主页的贵宾");
```

上面第二句和第三句的功能是一样的。

writeln()方法的功能与 write()相同，唯一的区别是，writeln()在输出字符串后再输出一个换行符。

（2）open()、close()、clear()方法

open()方法打开一个已存在的文件或创建一个新文件来写入内容，允许的文件类型（称为 mime 类型）包括 text/html、text/plain、image/gif、image/jpeg、image/xbm、x-world/plug-in 等，这些类型的文件是可以被浏览器解释或被插件支持的，默认的文件类型是 text/html，即标准 HTML 文件。text/plain 类型是纯文本文件。

close()方法是关闭文件；clear()方法是清理文件中的内容。

【例 5-17】Document 对象方法示例。

```
<html>
<head>
<script language="JavaScript">
function createWin(){
    NewWin=window.open("","","width=200,height=200");
    NewWin.document.open("text/html");
    NewWin.document.write("这是新创建的窗口!");
    NewWin.document.close();
}
</script></head>
<body>按下按钮可弹出一个窗口<br>
  <form><input type="button" value="弹出新窗口" onClick="createWin()"></form>
</body></html>
```

例 5-17 先在浏览器窗口中显示提示信息和一个按钮，用户单击按钮后，则弹出一个新窗口，新窗口显示的内容由 NewWin.document.write("这是新创建的窗口！")决定，该例在浏览器中的显示效果如图 5-17 所示。

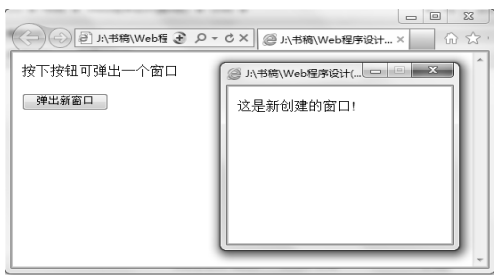


图 5-17　Document 对象方法示例

5.2.5　Form 对象

表单（Form）是 HTML 语言中动态更新页面内容的最常用的标记。3.2.4 节已经讨论了 Form 的语

法和输入域的类型，本节讨论 JavaScript 与 Form。

在 JavaScript 中，Form 也是对象，它封装了网页中由<form>标记定义的表单的信息。它是最复杂的 Navigator 对象。

1．Form 对象的属性

Form 对象的属性与<form>标记语法定义中的属性相对应，包括：

（1） action：表单提交后启动的服务器应用程序的 URL，与<form>标记定义中的 action 属性相对应；

（2） name：表单的名称，与<form>标记定义中的 name 属性相对应；

（3） method：指出浏览器将信息发送到由 action 属性指定的服务器的方法，它只可能是 get 或 post。Form 对象的此属性对应<form>标记定义中的 method 属性；

（4） target：指出服务器应用程序执行结果的返回窗口，对应<form>标记定义中的 target 属性；

（5） encoding：指出被发送的数据的编码方式，对应<form>标记定义中的 enctype 属性；

（6） elements：这是一个数组，其元素是表单的各个输入域对象，即 Form 对象的子对象，详情请参见本节第 3 点的说明；

（7） length：表单中输入域的个数。

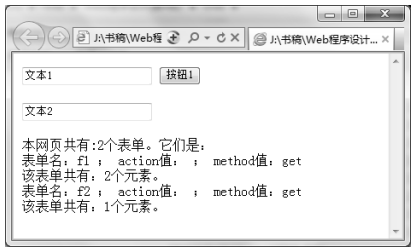


图 5-18　Form 对象属性引用方法示例

【例 5-18】Form 对象属性的引用方法示例。设计如图 5-18 所示的页面，放置两个表单，包含文本框和按钮，通过 Form 对象的属性和方法，在页面中显示表单元素。

```
<html><body><form name="f1">
<input type="text" name="t1" value="文本 1">
<input type="button" value="按钮 1"></form>
<form name="f2"><input type="text" name="t2" value="文本 2"></form>
<script language="JavaScript">
document.write("本网页共有:"+document.forms.length+"个表单。它们是：<br>");
for (var i=0;i<document.forms.length;i++){
        document.write("表单名："+document.forms[i].name+"  ；  ");
        document.write("action 值："+document.forms[i].action+"  ；  ");
        document.write("method 值："+document.forms[i].method+"<br>");
        document.write("该表单共有："+document.forms[i].length+"个元素。<br>");
}
</script></body></html>
```

2．Form 对象的方法

Form 对象的方法包括 submit()和 reset()。

submit()方法将触发 Submit()事件，引起 onSubmit 事件处理程序的执行。通常 Submit()事件被触发后，该表单中用户输入的数据将被提交给服务器端相应的程序；也可以通过给 onSubmit 事件处理程序返回 false 值来阻止数据被提交，利用这种功能可以实现对用户输入数据合法性的检查。

reset()方法清除表单中的所有输入，并将各输入域的值设为原来的默认值，该方法将触发 onReset 事件处理程序的执行。

Form 对象的这两个方法实际上模拟了 submit 和 reset 两个按钮的功能。

【例 5-19】用户输入数据合法性检查示例。设计如图 5-19 所示的页面，首先显示用户输入电话号码的界面，当用户输入了电话号码并单击“提交”按钮后，将执行合法性检查程序。若用户输入的电话号码值不符合要求（必须是数字，且数字位数为 11 位或 8 位），则提示出错信息，并且不提交服务器；否则将用户输入的电话号码数据提交给服务器。

图 5-19　数据合法性检查示例

```
<html><head><script language="JavaScript">
function Verify(){
   var Tel=document.TelForm.TelNo.value;
   if ((Tel.length==8) | (Tel.length==11)){
      if (parseInt(Tel)!=0){
         NewWin=window.open("","","width=200,height=200");
         NewWin.document.open("text/html");
         NewWin.document.write("<H3>号码已经成功提交！</H3>");
         NewWin.document.close();
      }
      else{
         alert("号码输入不正确！");
         return false;
      }
   }
   else{
      alert("号码输入不正确！");
      return false;
   }
}
</script></head>
<body>请输入您的电话号码：<br>
<form name="TelForm" onSubmit="Verify()">
<input type="text" name="TelNo" value=""><br>
<input type="submit" value="提交号码" width=100>
<input type="reset" value="重置号码" width=100></form>
</body></html>
```

3．Form 对象的子对象

从 Navigator 对象树形结构可以看出，Form 对象包含多个子对象，这些子对象对应 Form 的各种输入域。也就是说，Form 的每种输入域在 JavaScript 中都是作为对象来处理的。

（1）按钮对象

Form 的按钮有三种类型：submit—提交按钮；reset—复位按钮；button—普通按钮。

按钮包括如下属性：

```
name        //按钮名称
type        //按钮的类型
value       //按钮值，即按钮上显示的文字
weidth      //按钮的宽度
height      //按钮的高度
```

form　　//按钮所属于的表单

与按钮相关的事件与事件处理有：

Blur()事件和 onBlur 事件处理　　//失去焦点事件及处理

Focus()事件和 onFocus 事件处理　　//获得焦点事件及处理

Click()事件和 onClick 事件处理　　//单击 button 按钮的事件及处理

Submit()事件和 onSubmit 事件处理　　//单击 submit 按钮的事件及处理

Reset()事件和 onReset 事件处理　　//单击 reset 按钮的事件及处理

按钮对象的方法有 blur()、focus()、click()，它们将分别触发 onBlur、onFocus、onClick 事件处理程序。对于一个 submit 按钮，单击它与执行 Form 对象的 submit()方法是等价的；同样，对于一个 reset 按钮，单击它与执行 Form 对象的 reset()方法是等价的。

有关 Form 按钮子对象用法示例，读者可参见例 5-6（简易计算器模拟程序）。

（2）Form 对象的其他子对象

① text 对象（对应文本框 text），其属性有 name、type、form、value、defaultValue 等。name、type、value 属性的含义与按钮对象的同名属性相同；form 是包含该 text 的 form 的名称；defaultValue 属性是在网页被装入时文本框中显示的字符串。text 对象的方法有 focus()、blur()和 select()。前两个方法与 button 对象的同名方法相同；select()方法是文字框内的内容高亮度显示。与 text 对象有关的事件处理有 onFocus、onBlur、onChange、onSelect。text 对象的 Change()事件是在文字框的内容发生变化时被触发的，此时引起 onChange 事件处理程序的执行；用户在文字框内选择了内容，将触发 Select()事件，引起 onSelect 事件处理程序被执行。

② textarea 对象（对应文本域 textarea），textarea 对象和 text 对象有相同的属性和方法，它们的相关事件也相同。

③ password 对象（对应口令域 password），password 对象和 text 对象有相同的属性和方法，但注意它没有 OnClick 事件处理。

④ hidden 对象（对应隐藏域 hidden），它与 text 对象相比，没有 defaultValue 属性。

⑤ checkbox 对象（对应复选框 checkbox），它有 checked、defaultChecked、name、value、form、type 属性，其中 name、form、value 属性与其他 Form 子对象的同名属性含义相同；checked 属性是一个布尔值，反映当前复选框的状态，若 checked 为 true，则复选框被选中，否则未被选中；defaultChecked 属性也是一个布尔值，反映在复选框定义中是否有 checked 项，若有则 defaultChecked 属性值为 true，否则 defaultChecked 属性值为 false。checkbox 对象的方法有 blur()、focus()和 click()，相应的事件有 Blur()、Focus()、Click()，它们的含义与 button 对象完全相同。

⑥ radio 对象（对应单选钮 radio），它的属性、方法和事件与 checkbox 完全相同。

⑦ select 对象（对应选择列表 select），它拥有较多的属性和方法，还有 options 对象数组。其属性包括 form、name、length、type、selectedIndex 和 options。form、name 属性的含义与其他 Form 子对象同名属性相同；length 属性是 select 包含的选择项的个数；type 属性定义对象为 select 对象并指示 multiple 是否定义；selectedIndex 属性是被选中的选择项的索引号。options 数组包含了该 select 定义中的每个 option 定义的属性，其每个元素是一个 option 对象，option 对象对应 select 定义中的 option 选项定义。option 对象的属性包括 defaultSelected、index、selected、text、value。defaultSelected 属性反映 option 定义中有否 seleted 项，其值是一个布尔量；index 属性是一个整数值，等于该 option 选项的索引号，注意 select 定义中的索引号由 0 开始；selected 反映该选择项当前是否被选中，其值是一个布尔量；text 属性是选择项的文本，其值是一个字符串；value 对应 option 定义中的 value 属性设置。select 对象的方法有 blur()、focus()，相关的事件处理有 onBlur、onFocus、onChange。

下面给出一个示例说明 Form 子对象的用法。

【例 5-20】Form 子对象的用法示例。设计如图 5-20 所示的页面，填写和选择个人信息，由系统进行检查和显示。

```
<html><head><script language="JavaScript">
sex=new Array();
sex[0]="Male";
sex[1]="Female";
sele=0;
sex_sele=0;
function VerifyAndChgText(){
   var Length=document.forms[0].length;
   var Type,Empty=false;
   for (var i=0;i<Length;i++){
      Type=document.forms[0].elements[i].type;
   if (Type=="text")
      if (document.forms[0].elements[i].value=="")
         Empty=true;
   }
if (!Empty ){
   name="您的姓名是"+document.forms[0].NameText.value+"\n";
   alias="您的别名是"+document.forms[0].AliasText.value+"\n";
   sex_1="您的性别是"+sex[sex_sele]+"\n";
   area="您所在的地区是"+document.forms[0].area.options[sele].text+"\n";
   exp="备注信息是"+document.forms[0].exp.value+"\n";
   document.forms[0].info.value=name+alias+sex_1+area+exp;
}
   else
      alert("您未输入完全！");
   return Empty;
}
</script></head>
<body><h4 align=center>请输入您的个人信息</h4><form>
您的姓名：<input type=text name="NameText" size=10><br>
您的别名：<input type=text name="AliasText" size=10><br>
您的性别：<input type="radio" name="sex" onClick="sex_sele=0">Male
<input type="radio" name="sex" onClick="sex_sele=1">Female<br>
您所在地区：<select name="area" onChange="sele=this.selectedIndex">
<option value="1" selected>江苏省
<option value="2" >北京市
<option value="3" >上海市
<option value="4">天津市
<option value="5">浙江省
</select><br>
备注：<textarea name="exp" rows=3 cols=20></textarea><br><br>
<input type=button value="提交信息" onClick="VerifyAndChgText()">
<br>您已输入的信息是：<textarea name="info" rows=5 cols=30></textarea><br><br>
</form></body></html>
```

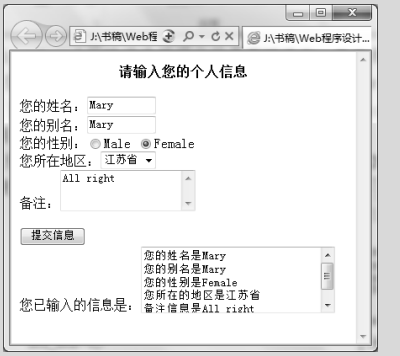


图 5-20　Form 子对象的用法示例

例 5-20 首先生成一个表单，要求用户输入个人信息，当用户单击“ 提交信息 ”按钮后，程序检查其是否输入完全，若输入完全，则在最后一个文本域中显示用户刚才输入的信息；若输入不完全，则给出警告提示。

程序中定义了数组 sex，存放 radio 各选项文本，用于显示；select 选择项的显示内容则引用

options 数组中 option 对象的 text 属性，并定义了两个数组的下标 sele 和 sex_sele。前者用做 select 的 options 数组下标，后者用做 sex 数组下标，它们的值分别通过 select 对象的 onChange、radio 的 onClick 事件处理获得。函数 VerifyAndChgText 检查用户是否在所有 text 中输入信息，若是，则将其输入和选择的信息在 info 文本域中显示；若否，则以 alert 给出警告，该函数作为“提交信息”按钮的 onClick 事件处理程序被执行。

5.2.6 History 对象和 Location 对象

1．History 对象

History 对象又称为历史清单对象，它是一个保存有窗口或帧在某个时间段内访问的 URL 信息的列表，并提供方法供用户在列表中查找。

History 对象包括如下属性。

（1）current：当前历史项的 URL。

（2）length：反映在历史列表中的项数。

（3）next：下一个历史项的 URL。

（4）previous：前一个历史项的 URL。

History 对象包括如下方法。

（1）back()方法装载历史列表中的前一个 URL。

（2）forword()方法装载历史列表中的下一个 URL。

（3）go()方法的参数可以是整数或字符串。当参数是整数 *i* 时，该方法将装载历史列表中与当前 URL 位置相距 *i* 的 URL，*i* 既可为正数，也可为负数。当参数是字符串时，该方法将装载历史列表中含该字符串的最近的 URL。

【例 5-21】在 HTML 文件中实现页面的前进和后退。

```
<html><head><title>History 对象示例</title></head>
<body><center><h2>History 对象使用示例</h2></center><hr><br>
<form><input type=button value="往前翻 " onClick="history.go(-1)">
<input type=button value="重载当前页" onClick="history.go(0)">
<input type=button value="往后翻 " onClick="history.go(1)"></form>
</body></html>
```

例 5-21 的运行结果如图 5-21 所示。

图 5-21　History 对象用法示例

2．Location 对象

Location 对象用于存储当前的 URL 信息，通过对该对象赋值来改变当前的 URL，例 5-14 已经使用过该对象的 hash 属性，通过改变 Location 的 hash 的值而改变了当前的页面。

Location 的属性与 Link 对象完全相同，包括 hash、host、hostname、href、pathname、port、protocol、search。

例如，下列两个语句将当前窗口的 URL 设置为 home.netscape.com：

```
window.location.href="http://home.netscape.com"
window.location="http://home.netscape.com"
```

5.2.7　Frame 对象

一个 Frame 对象对应一个<frame>标记定义。Frame 对象有如下几种属性。

（1）name：框架的名称，对应<frame>标记定义中的 name 项。

（2）length：框架中包含的子框架数目。

（3）parent：包含当前框架的 Window 或 Frame。

（4）self：代表当前框架。

（5）top：指包含框架定义的最顶层窗口。

（6）indow：与 self 含义相同。

（7）frames 数组：对应当前窗口中的所有框架。

Frame 对象的方法有 blur()、focus()、setTimeout()、clearTimeout()，它们与 Window 对象的方法完全一致。实际上，一个 Frame 对象就是一个特殊的 Window 对象，在框架情况下，self 和 window 属性就是当前的框架窗口。与 Frame 对象相关的事件处理是 onBlur、onFocus、onLoad、onUnload。

一个浏览器窗口可以包含多个框架，框架之间的信息传递可以加强框架的功能。例 5-22 说明了框架之间的通信。

【例 5-22】 框架之间的通信示例。

① 文件 frame 通信.htm，定义框架结构。

```
<html>
<frameset rows="15%,*">
    <frame src="frame 通信 1.htm" name="Win1">
    <frame src="frame 通信 2.htm" name="Win2">
</frameset>
</html>
```

② 文件 frame 通信 1.htm，是主要文件。

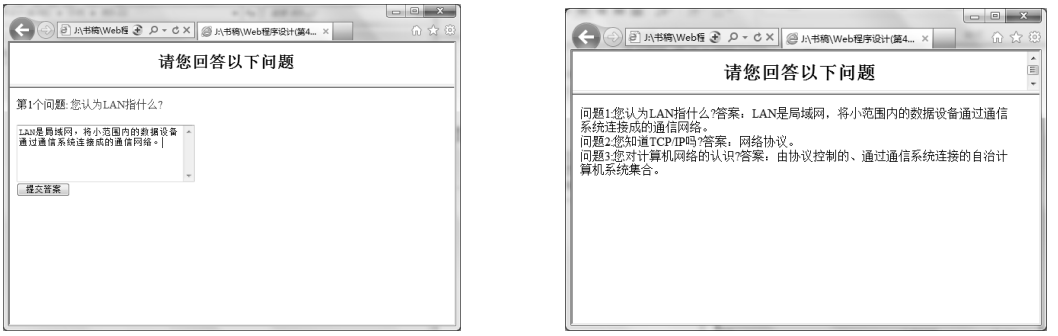
```
<html><head>
<script language="JavaScript">
var Questions=new Array("您认为 LAN 指什么?","您知道 TCP/IP 吗?","您对计算机网络的认识?");
var CurrentQuestion=0;
var Answers=new Array();
function PutQuestions(){ //在 Win2 中输出问题并创建表单，接收用户输入的答案
  with (parent.Win2.document){
    open();
    write("<html><body>");
    write("第",CurrentQuestion+1,"个问题: <font color=red>");
    write(Questions[CurrentQuestion]+"</font><br>");
    write("<form name=\'QuestionForm\'>");
    write("<textarea name=\'AnswerText\' cols=30 rows=5></textarea><br>");
    write("<input type=button value=\'提交答案\' onClick=\'parent.Win1.NextQ()\'>");
    write("</form></body></html>");
    close();
  }
}
function PrintAnswers(){ //在 Win2 中输出问题和用户输入的答案
  with (parent.Win2.document){
    open();
    write("<html><body>");
    for (var i=0;i<Questions.length;i++){
      write("问题",i+1,":",Questions[i]);
      write("答案：",Answers[i],"<br>");
```

```
        }
        write("</body></html>");
    }
}
function NextQ(){ //若问题未处理完，则调用 PutQuestions，否则，调用 PrintAnswers
if (CurrentQuestion<Questions.length){
    Answers[CurrentQuestion]=parent.Win2.document.QuestionForm.AnswerText.value;
    CurrentQuestion++;
    if (CurrentQuestion<Questions.length) PutQuestions();
    else { Answers[CurrentQuestion]=
            parent.Win2.document.QuestionForm.AnswerText.value;
        PrintAnswers();
    }}
}
</script></head>
<body onLoad="setTimeout('PutQuestions()',1000)">
<h2 align=center>请您回答以下问题</h2></body></html>
```

③ 文件 frame 通信 2.htm，可以为空，但为了浏览器的识别，写入了一些内容。

```
<html><head><title>文件 frame 通信 2.htm </title></head>
<body>This is a frame</body></html>
```

例 5-22 程序运行结果如图 5-22 所示。



（a）第一个问题　　（b）输出答案

图 5-22　框架之间通信的示例

frame 通信 1.htm 文件是对应于框架 Win1 的窗口文件，其中定义了两个数组 Questions 和 Answers，用于存放问题和用户提交的答案。该文件中的函数 PutQuestions()、NextQ()和 PrintAnswers()均是在框架 Win2 对应的窗口中输出文件，体现了在 JavaScript 中进行框架间通信的灵活性。该文件首先在框架 2 中提出第一个问题，在用户提交了第一个问题的答案后，再继续提问，直至所有问题都处理完，然后在框架 2 中显示各问题和用户相应的答案。

在框架间通信时，要注意框架窗口之间的层次关系。例如，对应 Win1 框架的 frame 通信 1.htm 文件要操作 Win2 时，需指出它们之间的关系，即 parent.Win2，表示 Win2 是 Win1 父窗口的子窗口。

下面通过两个实例，进一步深入讨论 HTML 与 JavaScript 程序设计技术。

5.2.8　程序示例 —— 用户注册信息合法性检查

用户通过表单将数据传递给服务器，如果将表单内的所有数据都交由服务器处理，则将加重服务器数据处理的负担。可利用 JavaScript 的交互能力，对用户输入的数据在客户端进行语法检查，然后把合法的数据传递给服务器。例 5-23 就是在用户填写表单并提交时，对用户所输入的数据在

客户端进行合法性检查的例子。

【例 5-23】设计如图 5-23 所示的用户注册页面。在单击“发送”按钮后，对输入框中输入的数据进行合法性检查。

图 5-23　利用 JavaScript 进行客户端输入数据检查

源代码如下：

```
<html>
<script language="JavaScript">
function init()
{ document.reg_form.usrname.focus();     //初始将光标定位在用户名输入框
}
function Verify()     //校验用户输入
{   if (VerifyUsrName()==false) return false;                    //校验用户名
    if (VerifyPasswd()==false) return false;                     //校验密码
    if (VerifyDepart()==false) return false;                     //校验单位名称
    if (VerifyAddr()==false) return false;                       //校验地址
    if (VerifyPersonName()==false) return false;                 //校验联系人姓名
    if (VerifyPhone()==false) return false;                      //校验电话号码
    if (VerifyZip()==false) return false;                        //校验邮编
    if (VerifyBp()==false) return false;                         //校验 Bp 号码
    if (VerifyFax()==false) return false;                        //校验传真号
    if (VerifyHand()==false) return false;                       //校验手机号
    if (VerifyEmail()==false) return false;                      //校验电子邮件地址
    if (VerifyHomepage()==false) return false;                   //校验主页地址
    if (VerifyQuest()==false) return false;                      //校验忘记密码时所提问题
    if (VerifyAnsw()==false) return false;                       //校验问题答案
    return true;
}
function VerifyUsrName()
{   if (document.reg_form.usrname.value.length==0)
    {     alert(" 用户名不能为空!请见左边的说明，输入合法的用户名。");
          return false;
    }
    if (validOfUsrName()==false)
```

```
    { alert("您输入的用户名中包含了不合法的字符!请见左边的说明，重新输入。");
      return false;
    }
    return true;
}
function validOfUsrName()      //检验用户输入的用户名中是否包含非法字符
{     valid=true;
      for (var i=0;i<document.reg_form.usrname.value.length;i++)
      {     var ch=document.reg_form.usrname.value.charAt(i);
            if (!((ch>="0") && (ch<="9")) && !((ch>="a") && (ch<="z")) && !((ch>="A") &&
                (ch<="Z")) && !(ch=="_"))
                valid=false;
            if (!valid)
                break;
      }
      return valid;
}
function VerifyPasswd()
{     if (document.reg_form.pass.value.length==0)
      { alert("  密码不能为空!请见左边的说明，输入您的密码。");
        return false;
      }
      if (document.reg_form.pass.value!=document.reg_form.pass2.value)
      { alert("您两次输入的密码不相同!请重新输入密码。");
        return false;
      }
       return true;
}
function VerifyDepart()
{     if (document.reg_form.dname.value.length==0)
      {     alert("单位名称不能为空!请见左边的说明进行输入。");
            return false;
      }
      return true;
}
//说明：校验联系人姓名、地址、邮编等与校验单位名称的正确性的程序相似，读者可自行补齐，
   其代码略去。
</script>
<body topmargin=6>
<center><font face="隶书" size=7 color=red>用户注册</font></center>
<hr width=90% color=blue><br>
<table border=0 aligen=center onLoad="init()">
<tr><td valign=top><!—对用户填写表单数据的说明-->
<table border=1 align=left bordercolor=blue bgcolor=ivory width=200
       style="fontsize:14px;align:left;" cellpadding=8><tr><td>
<div style="color:darkred;font-size:12px;text-align:left;font-family:'宋体'">
<span style="color:darkred;font-size:14px"><b> 说明</b></span><br><br>
<div>★为保证您今后在本网发布的供求信息的可靠性，请您如实填写会员信息表。
</div><br>
<div>★必须填写的基本信息：用户名、密码、忘记密码后查询密码
的问题、单位名称、联系人、地址、邮编、电话。</div><br>
<div>★<font color=black><b>用户名</b></font>:16 个字符以内的英
```

```
文字母数字串，可包含下画线"_"；</br>
★<font color=black><b>密码</b></font>：12 个字符以内的任意字符串。</div><br>
<div>★其他信息若您具备，最好也填写，以便于联系。</div><br>
</div></td></tr></table></td>
<td valign=top align=center><!—表单输入区定义-->
<form action="reg_handle.asp" method="get" name="reg_form"
onSubmit="return Verify()">
<table border=1 bordercolor=gold bgcolor=lavenderblush    width=550
style="font-size:12px;align:left;" cellpadding=4>
<tr><td width=550 ailgn=left colspan=2 >
 用户名  <input type=text size=16 name="usrname">
 密码  <input type=password size=12 name="pass">
 再输一遍密码  <input type=password size=12 name="pass2">
</td></tr>
<tr><td width=500    align=left colspan=2 >
 单位名称   <input type=text size=60 name="dname"></td></tr>
<tr><td width=550 colspan=2 align=left>
 联系地址   <input type=text size=60 name="address"></td></tr>
<tr><td width=550 align=left colspan=2>
 联系人姓名 <input type=text size=12 name="person_name">
 电话（区号+号码）<input type=text size=12 name="tel">
 邮编  <input type=text size=6 name="postcode"></td></tr>
<tr><td width=500 align=left colspan=2>
 寻呼  <input type=text size=14 name="bpcode">
 传真 <input type=text size=12 name="fax">
    移动电话  <input type=text size=12 name="hand"></td></tr>
<tr><td width=250 ailgn=left>
 E_mail <input type=text size=30 name="email"></td>
<td width=300 align=left> 主页<input type=text size=40 name="homepg">
</td></tr>
<tr><td width=550 colspan=2 align=left>
 忘记密码后查询时的问题   <input type=text size=50 name="quest">
</td></tr>
<tr><td width=550 colspan=2 align=left>
 忘记密码后查询问题的答案   <input type=text size=50 name="answ">
</td></tr></table><br>
<input type=submit value="发送">  <input type=reset value="重填">
</form></td></tr></table></body></html>
```

例 5-23 在浏览器中加载页面后，首先触发 onLoad 事件，执行 init()函数。init()函数将初始光标定位在用户名输入框。页面的总体结构是一个表，表又分为左、右两个部分，左部是对表单数据填写的说明，而右部则是表单输入区域。在表单属性中设置了 onSubmit 事件的处理函数 Verify()，Verify()函数分别再调用函数 VerifyUsrName()、VerifyPasswd()等 14 个函数检查用户名、密码等 14 个用户输入数据的合法性，若某个输入数据不合法，则以警告对话框提示用户重新输入。全部数据经过检查后，则认为用户输入的数据符合要求，函数 Verify()返回真值，即 onSubmit 事件处理返回真值，那么浏览器就开始发往服务器。否则 onSubmit 事件处理返回假值，那么数据不会提交给服务器处理。

5.2.9　程序示例 —— 扑克牌游戏程序

【例 5-24】读者一定很熟悉用扑克牌计算 24 点的游戏吧，下面就用 JavaScript 来设计一个在浏

览器中计算 24 点的游戏程序。游戏开始后，系统随机给出 4 个 1~9 之间的整数，用户给出这 4 个数的运算式，运算式中可包括加（+）、减（-）、乘（*）、除（/）运算符；若用户给出的算式的值为 24，则系统弹出一个对话框告之结果正确，否则告之结果错误。

本例共设计如下 5 个文件。

（1）game.htm：“ 开始新游戏 ” 界面，显示游戏规则，提供一个 “ 新游戏 ” 功能按钮，用户单击该按钮即可生成一个新窗口，在该窗口中加载 poker.htm 文件，开始一次新游戏的过程。

（2）poker.htm：主文件，它首先产生 4 个 1~9 之间的随机数，并将输入焦点定于结果算式输入框，然后等待用户输入；用户输入的时间限制为 1 min（60 000 ms），每隔 1 ms poker.htm 在状态栏提示用户所剩余的时间；若用户在规定的时间内在文本框中输入算式并单击 “ 提交答案 ” 按钮，poker.htm 将检查该答案，若计算正确，则弹出新窗口，其中加载文件 yes.htm，提示用户答案正确；若计算错误或算式不合法，也弹出新窗口，在其中加载文件 no.htm，提示用户答错了；若计时时间满了，而用户还未输入算式，也将弹出 “ 超时 ” 窗口，在其中加载文件 timeout.htm，提示用户超时，同时结束本次游戏。在该文件的 init()函数中，利用 Math 对象的数学函数 random()产生一个 0~1 之间的随机数；函数 ceil(number)将返回一个大于或等于 number 的最小整数，因此表达式 Math.ceil(Math.random()*9)将返回一个 1～9 随机之间的整数。该函数还利用 setTimeout()方法设置用户答题时间限制和状态栏提示信息的更新时间。文件中其他函数的功能及 HTML 语句的作用在下面的源文件中都给出了详细注解，在此不再赘述。

（3）yes.htm：当用户输入正确答案后弹出的提示窗口中要加载的文件。

（4）no.htm：当用户输入错误答案后弹出的提示窗口中要加载的文件。

（5）timeout.htm：超时后弹出的提示窗口中要加载的文件。

各文件的内容如下。

① 文件 game.htm

```
<html><head><title>扑克牌游戏</title>
<style type="text/css">
body {margin-top:16;}
h3{ background-color:#ffddcc;color:darkred;width:16%;height:60px;
vertical-align:50%;}
</style></head><body><h3><br>  游戏规则：</h3><br>
<p><h4>规则 1. 要进行新游戏，请按 “ 新游戏 ” 按钮，此时将生成一个新窗口，其中给出 4 个
    1~9 之间的整数。
<br>规则 2. 在新窗口的输入框输入四则运算式，可以使用括号，运算数为给出的 4 个值，每个
    数只能使用 1 次，结果需等于 24。
<br>规则 3. 若您认为给出的 4 个数的运算结果不可能为 24，则可按 “ 放弃这局 ” 按钮。
</h4></p>
<form>
<input type="button" value="新游戏"
        onClick="open('poker.htm','newWin','width=500,height=400,status=1')">
</form></body></html>
```

② 文件 poker.htm

```
<html><head>
<script language="JavaScript">
//全局变量定义
var card = new Array();              //存储 4 张牌
for (k=0;k<4;k++)
    card[k]=0;                       //赋初值
var cardUsed = new Array();          //每张牌是否被使用过的标记
for (k=0;k<4;k++)
```

```
        cardUsed[k] = false;              //赋初值
var TimeID,StatID;                        //时间和状态标记
var count = 60;                           //计时器
function Init() {                         //初始化函数，生成 4 个随机数，并设置时间和状态标记
var i;
status = "您有 1min 的时间考虑与输入答案！";
for (i=0;i<4;i++)                         //用随机函数产生 4 张牌
        card[i] = Math.ceil(Math.random()*9);
StatID = setTimeout("ChangeStatus()",1000);
TimeID = setTimeout("open('timeout.htm','timeoutWin',
        'width=200,height=100');close()",60000);
}     //End of Init
//状态栏刷新函数定义
function ChangeStatus() {                                  //每隔 1s 刷新 1 次状态栏显示
clearTimeout(StatID);                                      //清除状态标记
count--;                                                   //剩余时间减少 1s
status = "剩余时间为："+count+"s";
setTimeout("ChangeStatus()",1000);                         //每隔 1s 调用 1 次 ChangeStatus
}
//输入合法性判断函数，若算式合法，返回 true，否则返回 false
function IsValid() {                                       //判定用户输入的算式是否正确
var exp = document.expForm.expText.value;                  //取用户输入的算式
var expLen = exp.length;                                   //算式长度
var i,j;
var numberUsed = 0;                                        //算式中使用的运算数的个数
for (i=0;i<expLen-1;i++){
        var ch = exp.charAt(i);                            //取第 i 个字符
        if (ch>='0' && ch<='9') {                          //当前处理的是数值字符
        for (j=0;j<4;j++)
                if ((ch == card[j]) && (!cardUsed[j])) {
                    //该数字是否是给出的 4 个数之一且未被使用过
                numberUsed++;
                cardUsed[j] = true;                        //置数字已被使用过标记
                }
        }
        else {    //当前处理的是运算符
                  if ((ch!="+") && (ch!="-") && (ch!="*") && (ch!="/") && (ch!="(")
                          && (ch!=")"))
                   {  alert("您输入的算式是非法的!");
                      return false;
                   }
        }
}
if (numberUsed!=4) {    //算式中未使用全部 4 个数字
        alert("您输入的算式是非法的!");
        return false;
}
return true;
}
function calResult() { //计算算式结果函数
         clearTimeout(TimeID);    //清除计时标记
         if (IsValid()) {    //算式合法
                if (eval(document.expForm.expText.value)==24)   //若算式的值等于 24
```

```
                {   //弹出新窗口，提示结果正确
                    winid = open("yes.htm","nwin1","width=200,height=100");
                    close();
                    return;
                }
            }
            //算式的值不等于 24 或算式不合法，弹出新窗口，提示结果错误
            winid = open("no.htm","nwin2","width=200,height=100");
            close();
            return;
        }
        </script></head>
        <!—将输入焦点设置在答案输入框中-->
        <body onLoad="document.expForm.expText.focus()">
        <script language="JavaScript">
        Init(); //生成四张牌
        document.write("您可以使用的四张牌是：<br>");
        for (i=0;i<4;i++)
            document.write(card[i]+"    ");
        </script>
        <br><form name="expForm">请在右边的输入框中输入您的答案：  
        <input type=text name="expText" size=12 value=" "><br><br><br>  
        <input type=button value="提交答案" onClick="calResult()">
        <input type=button value="放弃该局" onClick="self.close()"></form></body></html>
```

③ 文件 yes.htm

```
<html>
<body onLoad="setTimeout('close()',60000);return true">恭喜您，您答对了！</body>
</html>
```

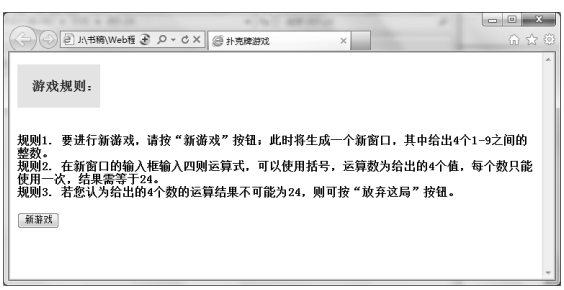
④ 文件 no.htm

```
<html>
<body onLoad="setTimeout('close()',60000);return true">对不起，您答错了！</body>
</html>
```

⑤ 文件 timeout.htm

```
<html>
<body onLoad="setTimeout('close()',60000);return true">
对不起，您超时了！请重新开始。</body></html>
```

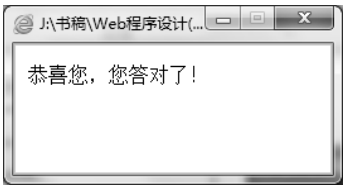
该程序的运行过程是，首先加载文件 game.htm，将出现如图 5-24（a）所示的界面，然后单击“新游戏”按钮，弹出如图 5-24（b）所示的游戏窗口，在答案输入框中输入算式，并单击“提交答案”按钮；若答案正确，将出现图 5-24（c）所示的提示窗口；若答案不正确或超时，所出现的提示窗口与图 5-24（c）类似。



（a）开始一次“新游戏”界面

图 5-24　扑克牌游戏示例

（b）“游戏”界面



（c）答案正确的提示窗口

图 5-24　扑克牌游戏示例（续）

5.3　HTML DOM

5.3.1　HTML DOM 概述

DOM，即 Document Object Model，文档对象模型，它是由 W3C 提出的。W3C 于 1998 年 10 月推出 DOM Level 1，又于 2000 年 11 月推出了规范 DOM Level 2。DOM 是一个跨平台的、可适应不同程序语言的文件对象模型，它采取直观且一致的方式，将 HTML 或 XML 文档进行模型化处理，提供存取和更新文件内容、结构和样式的编程接口。使用 DOM 技术，不仅能够访问和更新页面的内容及结构，而且还能操纵文件的风格样式。

DOM 是从 DHTML 对象模型发展而来的，它是对 DHTML 对象模型进行了根本变革的产物。使用 DHTML 对象模型技术，能够单独地访问并更新 HTML 页面上的对象，每个 HTML 标记通过它的 id 和 name 属性被操纵，每个对象都具有自己的属性、方法和事件，通过方法操纵对象，通过事件触发因果过程。DOM 则比 DTHML 对象模型功能更全面，它提供一个对整个文件的访问模型，而不再仅仅局限于单一的 HTML 标记范围内。DOM 将文件作为一个树形结构，树的每个节点表现为一个 HTML 标记或 HTML 标记内的文本项。树形结构精确地描述了 HTML 文件中标记间及文本项间的相互关联性，这种关联性包括 child（孩子）类型、parent（双亲）类型和兄弟（sibling）类型。

DOM 将 HTML 或 XML 文件转换为内部树形结构，使程序设计者能够更容易地处理文件的内容，其优点是：① 平台无关性，DOM 提供跨平台的编程接口，是一种处理 HTML 和 XML 文件的标准 API；② 可支持对 HTML 及 XML 两种文件的处理。

5.3.2　DOM 节点树

DOM 是一种结构化的对象模型，采用 DOM 技术访问和更新 HTML（或 XML）页面内容时，首先依据 HTML（或 XML）源代码，推出页面的树形结构模型，然后按照树形结构的层次关系来操纵需要的属性。例如，要更新页面上的文本项内容，如果采用 DTHML 对象模型，需要使用 innerHTML 属性，但必须要注意，并不是所有的 HTML 对象都支持 innerHTML 属性；而如果采用 DOM 技术，则只要修改相关树节点都具有的 nodeValue 属性值即可。

【例 5-25】产生表格的 HTML 文件示例。DOM 将该文件作为如图 5-25 所示的树形结构。

```
<table>
<tbody>
    <tr>
        <td>商品类别</td>
        <td>数量</td>
```

```
        </tr>
        <tr>
            <td>日用百货</td>
            <td>10</td>
        </tr>
        <tr>
            <td>电器</td>
            <td>20</td>
        </tr>
    </tbody>
    </table>
```

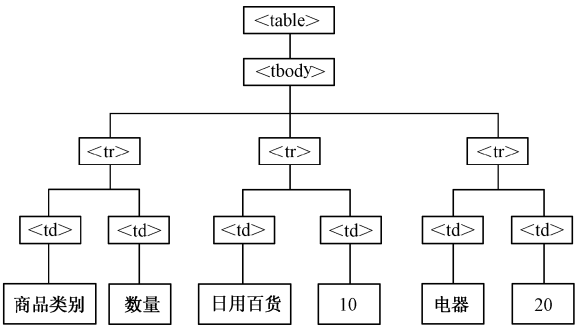


图 5-25　例 5-25 对应的 DOM 树形结构

文件的 DOM 树形结构表示 HTML 文件各对象的关系。在 DOM 树形结构中，每个节点都是一个对象，各节点对象都有属性和方法。树中的叶节点为文字节点，页面显示的内容就是各文字节点的内容。

5.3.3　DOM 树节点的属性

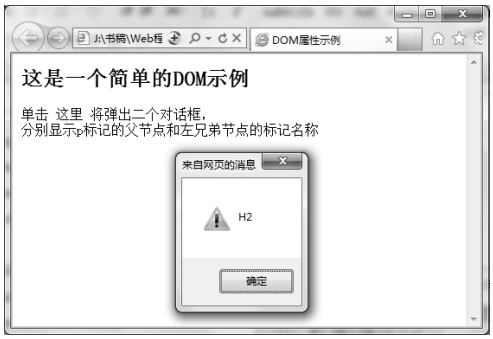
DOM 树形结构的节点有只读属性和读/写属性两类，通过只读属性可以浏览节点，并可获得节点的类型及名称等信息；通过读/写属性可以访问文字节点的内容。DOM 树节点的属性列于表 5-11 中。

表 5-11　DOM 树节点的属性

属　性	访　问	说　明
nodeName	只读	返回节点的标记名
nodeType	只读	返回节点的类型：1—标记；2—属性；3—文字
firstChild	只读	返回第一个子节点的对象集合
lastChild	只读	返回最后一个子节点的对象集合
parentNode	只读	返回父节点对象
previousSibling	只读	返回左兄弟节点对象
nextSibling	只读	返回右兄弟节点对象
data	读/写	文字节点的内容，其他节点返回 undefined
nodeValue	读/写	文字节点的内容，其他节点返回 null

【例 5-26】在下列 JavaScript 程序中，分别用 parentNode 和 previousSibling 获得父节点和左兄弟节点，用 nodeName 属性输出节点的标记名。本例在浏览器中的运行结果如图 5-26（a）、（b）所示。

（a）单击<p>区域，弹出的对话框显示其父节点名



（b）单击(a)中对话框“ 确定 ”按钮弹出显示其左兄弟节点名的对话框

图 5-26　例 5-26 的运行结果

```
<html>
<head><title>DOM 属性示例</title>
<script language="JavaScript">
function Access()
{ ShowParentNode();
   ShowLeftSiblingNode();
}
function ShowParentNode()
{   var pnode=p1.parentNode;
    alert(pnode.nodeName);
}
function ShowLeftSiblingNode()
{   var prenode=p1.previousSibling;
    alert(prenode.nodeName);
}
</script>
</head>
<body>
<h2>这是一个简单的 DOM 示例</h2>
<p id="p1" onClick="Access()">
单击这里将弹出二个对话框，<br>
分别显示 P 标记的父节点和左兄弟节点
  的标记名称<br>
      </p>
</body>
</html>
```

对该文件的分析可知，标记<p>的父节点是<body>，左兄弟节点是<h2>。

DOM 有两个对象集合：attributes 和 chileNodes。

① attributes 是节点内容的对象集合。

② chiledNodes 是子节点的对象集合，可使用从 0 开始的索引值进行访问。

可以对文字节点使用 data 或 nodeValue 属性访问和修改节点的内容，常用的是 nodeValue 属性。

5.3.4　访问 DOM 节点

用 DOM 的方法可以创建 HTML 或 XML 文件，并可以通过 JavaScript 程序随时改变文件的节点结构或内容，建立动态网页效果。表 5-12 列出了 DOM 的方法。

表 5-12　DOM 的方法列表

方　法　名	说　　明
appendChild objParent.appendChild(objChild)	为 objParent 节点添加一个子节点 objChild，返回新增的节点对象
applyElement objChild.applyElement(objParent)	将 objChild 新增为 objParent 的子节点
clearAttributes objNode.clearAttributes()	清除 objNode 节点的所有属性
createElement document.createElement(“TagName”)	建立一个 HTML 节点对象，参数 TagName 为标记的名称
createTextNode document.createTextNode(String)	建立一个文字节点，参数 String 是节点的文字内容
cloneNode objNode.cloneNode(deep)	复制 objNode，参数 deep 若为 false，则仅复制该节点；否则，复制以该节点为根的整棵树
hasChildNodes objNode.hasChildNodes()	判断 objNode 是否有子节点，若有则返回 true，否则返回 false
insertBefore objParent.insertBefore(objChild,objBrother)	在 objParent 节点的子节点 objBrother 之前插入一个新的子节点 objChild
mergeAttributes objTarget=mergeAttributes(objSource)	将节点 objSource 的属性复制合并到节点 objTarget 中
removeNode objNode.removeNode(deep)	删除节点 objNode，若 deep 为 false，则只删除该节点；否则，删除以该节点为根的子树
replaceNode objNode.replaceNode(objNew)	用节点 objNew 替换节点 objNode
swapNode objNode1.swapNode(objNode2)	交换节点 objNode1 与 objNode2

【例 5-27】使用 DOM 的方法生成一个表格。它在浏览器中的运行结果如图 5-27（a）、（b）所示。

```
<html>
<head><title>使用 DOM 生成表格</title>
<script language="JavaScript">
function genTable(pNode)
{
   var i,j;
   var contents=new Array(3);
   for (i=0;i<3;i++)
         contents[i]=new Array(2);
   contents[0][0]="商品类别";
   contents[0][1]="数量";
   contents[1][0]="日用百货";
   contents[1][1]="10";
   contents[2][0]="电器";
   contents[2][1]="20";
   var tableNode=document.createElement("TABLE");
   var tBodyNode=document.createElement("TBODY");
   var t1,t2;
   for (i=0;i<3;i++)
   {
```

```
        t1=document.createElement("TR");
        tBodyNode.appendChild(t1);
        for (j=0;j<2;j++)
           {
             t1=document.createElement("TD");
             t2=document.createTextNode(contents[i][j]);
             t1.appendChild(t2);
             tBodyNode.childNodes[i].appendChild(t1);
           }
      }
      pNode.appendChild(tableNode);
      tableNode.id="test";
      tableNode.border=2;
      tableNode.appendChild(tBodyNode);
}
</script>
</head>
<body id="tableTest">
<h2 onClick="genTable(tableTest)">单击此处将生成一个表格</h2><hr>
</body>
</html>
```

（a）初始显示　　（b）单击文字后，生成一个表格

图 5-27　例 5-27 的显示结果

DOM 可以操纵文档的树形结构，包括创建新节点、删除存在的节点或在树形结构中移动节点等。而 DTHML 对象模型则不允许更改文档结构，只能操纵现有的对象。这一点是 DOM 对 DHTML 对象模型最本质的改进。

本 章 小 结

本章讨论了 Web 客户端程序设计的两种关键技术：脚本语言和对象模型。

首先讨论了脚本语言的特点，然后着重讲解 JavaScript 语言。JavaScript 是基于对象的，具有很好的跨平台特性，适用于大多数浏览器，其基本语法类似于 C 语言。此外，JavaScript 还定义了较丰富的对象和函数，使其处理能力得到增强。说明了 JavaScript 的基本语法规则、JavaScript 预定义对象及函数的使用方法。

浏览器对象模型将网页处理为对象的集合，网页元素都可以是对象，具有属性、方法和事件，通过脚本语言就可以操作网页元素。本章详细阐述了 Navigator、Window、Document、Form、History 和 Frame 等浏览器对象的基本功能和使用方法，以大量的示例介绍了使用 JavaScript 语言进行浏览器对象编程的技术。本章还简要说明了通过 HTML DOM 进行网页内容访问与控制的方法。

习 题 5

5.1 简述脚本语言的特点。

5.2 什么是对象？什么是事件？

5.3 JavaScript 的对象分为哪两类，各有什么特点？

5.4 JavaScript 中如何创建对象？

5.5 试用 JavaScript 语言设计一个程序，判断用户输入的整数是正数、负数还是零。

5.6 试设计一个程序，判定用户输入的电话号码是否正确，假设电话号码可以是 7、8 或 11 位。

5.7 试设计一个程序，根据当天是星期几，在页面中显示不同的图片。

5.8 简述浏览器对象模型的含义。

5.9 浏览器对象模型中包含哪些主要对象？

5.10 Navigator 对象有哪些常用属性和方法？

5.11 Window 对象有哪些常用属性和方法？

5.12 Document 对象有哪些常用属性和方法？

5.13 Form 对象有哪些常用属性和方法？

5.14 Form 对象有哪些子对象？它们有什么特点？

5.15 如何通过浏览器对象实现页面框架间的通信？

5.16 如何通过浏览器对象实现对用户输入数据的正确性验证？这样做有什么优点？

5.17 简述 DOM 的含义。

上机实验 5

实验 5.1 用 JavaScript 脚本语言设计条件判断程序。

【目的】

（1）掌握将 JavaScript 脚本嵌入 HTML 文件的方法。

（2）掌握使用 JavaScript 脚本语言设计应用程序的过程和基本的 JavaScript 语法。

【内容】用 JavaScript 脚本语言设计一个程序：根据当天是星期几，在页面中显示不同的图片。程序的运行结果如图 5-28 所示。要求在图片上方显示今天是星期几，再显示图片。

图 5-28 根据星期几显示不同的图片

【步骤】

（1）准备 7 个 jpg 图片文件，分别命名为 1. jpg～7. jpg。

（2）打开记事本程序。

（3）输入能够生成如图 5-28 所示页面的 JavaScript 程序的源代码，保存为.html 文件，文件名为 ex5-1。

（4）双击 ex5-1.html 文件，在浏览器中查看结果。

实验 5.2　用 JavaScript 脚本语言实现客户端输入数据的正确性检查。

【目的】

（1）掌握表单的使用方法。

（2）掌握使用 JavaScript 脚本语言对客户端输入数据的操作方法。

【内容】在例 5-23 所给出的信息基础上，完成所有的程序，并进行测试。

【步骤】

（1）打开记事本程序。

（2）输入能够生成如图 5-23 所示页面的 html 文件，并嵌入实现客户端输入正确性检查的 JavaScript 程序源代码，保存为 ex5-2.html 文件。

（3）双击 ex5-2.html 文件，在浏览器中查看结果。

实验 5.3　用 JavaScript 脚本语言实现浏览器对象模型编程。

【目的】

（1）掌握浏览器对象模型。

（2）掌握使用 JavaScript 脚本语言对浏览器对象模型编程。

【内容】用 JavaScript 脚本语言设计一个程序：在页面中显示一个按钮“显示定时的警告框”，点击该按钮，经过 10 秒钟后，弹出警示框。程序的运行结果如图 5-29 所示。

【步骤】

（1）打开记事本程序。

（2）输入能够生成如图 5-29 所示页面的 html 文件，并嵌入实现浏览器对象模型操作的 JavaScript 程序源代码，保存为 ex5-3.html 文件。

（3）双击 ex5-3.html 文件，在浏览器中查看结果。

图 5-29　定时弹出警示框

第 6 章　Web 服务器端程序设计

ASP.NET 是 Microsoft 公司推出的一个统一的 Web 开发模型，是.NET Framework 体系结构的重要组成部分，也是目前开发 Web 应用程序的两大主流技术之一。它采用面向对象、事件驱动的编程技术，在性能和效率上全面超越 ASP，一经推出就备受关注。经过多年的改进和优化，已逐渐成为成熟、稳定、功能强大的服务器端编程环境。

6.1　初识 ASP.NET

6.1.1　一个简单的 ASP.NET 程序 —— 用户登录程序

下面是一个简单的 ASP.NET 程序。通过这个实例，可以大致地了解 ASP.NET 程序的架构和编程语法。

【例 6-1】一个用户登录程序，分为学生、教师、管理员三类用户。输入用户名和密码以后，根据用户的身份分别显示不同的欢迎词。程序运行结果如图 6-1 所示。

源程序代码如下：

```
<%@ Page Language="C#"    AutoEventWireup="true"%>
<script runat="server">
void Button1_Click(object sender, EventArgs e)
{
        if (usename.Text == "administrator" && Radioteacher.Checked == true &&    usepassword.Text != "")
        //输出身份是管理员的欢迎词
            Response.Write("欢迎你管理员同志！");
        else if (Radioteacher.Checked == true && usepassword.Text != "" )
            Response.Write("欢迎你" + usename.Text + "老师！");   //身份是老师
        else if (usepassword.Text != "" )
            Response.Write("欢迎你"+usename.Text +"同学！");     //身份是学生
 }
void Button2_Click(object sender, EventArgs e)
{           usename.Text ="";
            usepassword.Text="";
            usename.Focus();
}
</script>
<html>
<head runat="server">
    <title>无标题页</title>
</head>
<body>
    <form id="form2" runat="server">
     用户登录
    <hr/>
<%--下面构造一个表单--%>
    <div>
```

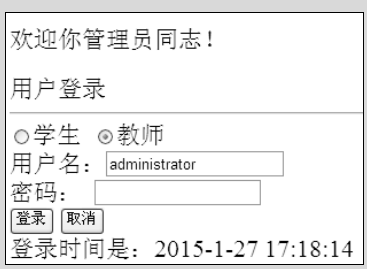


图 6-1　例 6-1 运行结果

```
                    <!--RadioButton 单选钮  -->
                    <asp:RadioButton ID="Radiostudent" runat="server" checked="true"
                    GroupName="sel" Text="学生" />
                <asp:RadioButton ID="Radioteacher" runat="server" GroupName="sel" Text="教师" />
                <br />
                 <!--TextBox 单行文本输入框-->
                 用户名：<asp:TextBox ID="usename" runat="server"></asp:TextBox>
                <br />
                  <!--TextBox 密码文本输入框-->
                 密码：
                <asp:TextBox ID="usepassword" runat="server" TextMode="Password"></asp:TextBox>
                <br />
                <asp:Button ID="Button1" runat="server"    onclick="Button1_Click" Text="登录" />
                <asp:Button ID="Button2" runat="server" onclick="Button2_Click" Text="取消" />
            </div>
           <!--输出当前日期和时间-->
            登录时间是：<%=DateTime.Now%>
           </form>
    </body>
    </html>
```

说明：以上程序可以用文本编辑器（Notepad）或其他编辑器输入，并保存在 Web 服务器的虚拟目录下。在浏览器的地址栏中输入 http://localhost/6-1.aspx，就可以看到如图 6-1 所示的运行结果。

6.1.2　ASP.NET 程序结构分析

ASP.NET 程序文件是一个扩展名为.aspx 的文本文件。当客户端请求到来时，Web 服务器将请求提交给 ASP.NET 模块处理，在服务器上动态编译和执行，产生一个 HTML 流，然后传送给发出请求的客户端浏览器。

1．页面的基本元素和语法

从例 6-1 中看到，一个 ASP.NET 的页面可以包含以下多种元素：页面编译指令、代码声明块、代码呈现块、代码注释、ASP.NET 控件、文本和 HTML 标记、服务器端包含指令。

（1）页面编译指令

页面编译指令是供编译器处理 ASP.NET 页面和用户控件时使用的命令，可以放在页面的任何位置，作为惯例，通常将它放在 ASP.NET 文件的开头，如例 6-1 中的第一行：

```
<%@ Page Language="C#" AutoEventWireup="true" %>
```

页面编译指令的语法格式如下：

```
<@% 指令名 属性＝属性值 %>
```

当然，页面编译指令不是文件中必须的。在.aspx 文件中常用的页面编译指令有以下几种。

@Page　　配置页面被处理和编译时与之相关的属性。

@Import　　将名称空间导入到当前页面中。

@Register　　允许注册其他控件以便在页面上使用。

@Assembly　　在编译时将程序集链接到页面，使程序员可以使用程序集公开的所有类和方法。

@Implements　　定义要在页或用户控件中实现的接口。

① @Page 指令。它是最常用的一个页面编译指令。每个 aspx 文件只包含一个@page 指令，定义多个属性时，以空格分开。

@Page 指令的语法格式如下：

```
<%@ Page  属性名="属性值" [属性名="属性值"] %>
```

@Page 指令常用的属性如表 6-1 所示。

表 6-1　@Page 指令的常用属性

属　　性	说　　明
AutoEventWireup	取值 true 或 false。指定页面的事件是否自动触发。默认值为 true，自动传送
Buffer	指定是否启用 HTTP 相应缓冲区。默认值为 true，启用缓冲区
CodeBehind	指定与页面相关的后台代码文件名
Errorpage	用于在出现未处理页异常时，重定向目标 URL
Explicit	指定页面是否使用 Visual Basic Option Explicit 模式进行编译。默认值为 true，表示所有的变量都必须先定义后使用
Inherits	定义页要继承的基类，可以是从 Page 派生而来的任何类
Language	指定编程逻辑中使用的程序设计语言。可以是任何一种.NET 支持的程序设计语言，如 VB、C#等
MasterPageFile	设置内容页的母版页或嵌套母版页的路径。支持相对路径和绝对路径

例如：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="login.aspx.cs"
Inherits="WebApplication1.login" %>
```

@Page 指令是页面默认指令，可以省略 Page，直接写成：

```
<%@ Language="C#" AutoEventWireup="true" CodeBehind="login.aspx.cs"
Inherits="WebApplication1.login" %>
```

② @Import 指令。将名称空间导入当前页面中，这样页面便可以使用该名称空间中定义的类和接口。被导入的名称空间可以是.NET 框架类库或用户定义的其他名称空间。

@Import 指令的语法格式如下：

```
<@Import Namespace="value" %>
```

其中，value 为要引入的名称空间。每条@Import 指令只能引入一个名称空间。若要引入多个名称空间，需使用多条@ Import 指令。例如：

```
<%@ Import Namespace="System.Data" %>
<%@ Import Namespace="System.Data.OleDb" %>
```

以上表示在 ASP.NET 网页中引入了两个名称空间。接下来我们要申明两个变量，此变量属于已引用的两个名称空间下的类，如：

```
OleDbConnection MyConnection = new OleDbConnection();
OleDbCommand MyCommand = new OleDbCommand();
```

说明：OleDbConnection 及 OleDbCommand 都是 System.Data.OleDb 下的类。

（2）代码声明块

定义一段在服务器上运行的程序代码，用来生成动态的 Web 页面，一般写在程序的开始部分。语法格式如下：

```
<script   language＝"编程语言" runat="server">
     代码
</script>
```

其中，属性 language 的值可以是.NET 支持的任何一种编程语言，如 VB.NET、C#、Jscript.NET 等。如果没有指定，则采用@Page 指令中配置的语言。若@Page 指令中也没有定义，默认是 VB.NET。例 6-1 中<script runat="server">，表明使用@Page 指令中配置的 C#语言。runat 属性值 server，表示该段代码在服务器上执行。若不设置该属性，则该段程序由客户端的浏览器执行。

（3）代码呈现块

定义呈现网页时所执行的内联代码或内联表达式。语法格式如下：

```
<% 内联代码 %>
<%=内联表达式%>
```

如例 6-1 中的内联表达式：

```
<%=DateTime.Now%>
```

代码呈现块在 ASP 中至关重要，而在 ASP.NET 中已被更好的机制代码声明块所取代。

注意：<%…%>标记中不能编写事件处理过程。

（4）代码注释

注释是程序代码不可缺少的部分，注释的目的是帮助开发人员和其他人员理解程序代码。注释元素开始标记和结束标记中的内容在执行时既不会被服务器处理，也不会交付给结果页面显示。ASP.NET 文件中的注释有三种形式：HTML 注释、代码注释和服务器端注释标记。

① HTML 注释。语法格式如下：

```
<!--注释-->
```

如例 6-1 中的 HTML 注释：

```
<!--TextBox 密码文本输入框-->
```

② 服务器端注释标记。语法格式如下：

```
<%--注释--%>
```

如例 6-1 中的服务器端注释标记：

```
<%--下面构造一个表单--%>
```

③ 代码注释。一般来说，ASP.NET 程序的绝大多数地方都可以使用服务器端注释标记<%--注释--%>，但在代码声明块和代码呈现块中通常习惯使用编程语言的注释标记。语法格式如下：

```
<script    language＝"C#" runat="server">
              代码
            /*
                 注释块
              */
  //行注释
</script>
```

或者：

```
<script    language＝"VB" runat="server">
        代码
        '注释
</script>
```

在例 6-1 中有如下 C#语言的行注释标记：

```
//输出身份是管理员的欢迎词
```

注意：如果在代码呈现块<%…%>中使用服务器端注释，将会出现编译错误。

（5）ASP.NET 控件

ASP.NET 的控件有许多，主要有 HTML 服务器控件和 Web 服务器控件，是构成用户界面的重要元素。其中 HTML 服务器控件从 HTML 标记发展而来的，增加了 id 属性和 runat 属性，运行于服务器端。例如：

```
<input type="button" id="Submit1" value="登录" runat="server" onServerClick="b_click" />
```

Web 服务器控件除了具有 HTML 控件的属性外，还有方法和事件，比 HTML 控件的功能更强大。如例 6-1 中使用的 Web 服务器控件 Button1 以及它的事件过程 Button1_Click。

```
<asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="登录" />
void Button1_Click(object sender, EventArgs e)
{
        if (usename.Text == "administrator" && Radioteacher.Checked == true
&&   usepassword.Text != "")
```

```
                    //输出身份是管理员的欢迎词
                    Response.Write("欢迎你管理员同志！");
                else if (Radioteacher.Checked == true && usepassword.Text != "" )
                    Response.Write("欢迎你" + usename.Text + "老师！"); //身份是老师
                else if (usepassword.Text != "" )
                    Response.Write("欢迎你"+usename.Text +"同学！"); //身份是学生
        }
```

需要注意的是，ASP.NET 服务器控件必须放置在<form runat="server"></form>标记之间，并标记为 runat="server"，如例 6-1 中使用的各种 Web 服务器控件。其具体内容将在 6.3.3 节中讨论。

除了上述两种控件外，用户还可以定义自己的控件，并将其加入到页面中使用。

（6）文本和 HTML 标记

如例 6-1 中的文本“ 用户登录 ”和众多的 HTML 标记<hr/>、
、<form>等。

（7）服务器端包含指令

服务器端包含指令用于将指定文件的内容插入到 ASP.NET 页内或用户自定义的控件中，其作用相当于将两个文件合并成一个文件。被插入的文件可以是网页文件（.aspx）、用户控件文件（.ascx）和 Global.asax 文件。语法格式如下：

```
<!--#include file|virtual=filename-->
```

其中 file 关键字表示使用包含文件在服务器上的物理路径，可以是绝对路径或相对路径，但必须与页面文件在同一路径下。

virtual 关键字表示使用包含文件在网站的虚拟路径。和 file 一样，也可以是绝对路径或相对路径。

filename 是 file 或 virtual 的属性值，是想包含的文件的路径和名称，用双引号括起来。

例如，如果一个被命名为 footer.inc 的文件属于一个名为 /myapp 的虚拟目录，则下面的语句将把 footer.inc 的内容插入到包含该行指令的文件中：

```
<!--#include virtual ="/myapp/footer.inc"-->
```

2．ASP.NET 的页面模式

ASP.NET 页面由两部分组成：可视元素和编程逻辑。可视元素由 HTML 标记、静态文本和 ASP.NET 服务器控件构成，以<HTML>标记开始，</HTML>标记结束，用于实现 Web 应用程序与用户交互的界面；编程逻辑由程序设计语言编写的代码构成，介于标记<Script runat="server">和</Script>之间，完成 Web 应用程序的功能。过去，ASP 程序设计中采用的是可视元素和编程逻辑混合在一个.asp 文件中的模式。现在，ASP.NET 提供两种管理模式，分别是单文件页模式和代码隐藏页模式，两种模式的功能完全相同。

（1）单文件页模式，即过去的 ASP 模式。它将可视元素和编程逻辑放在同一个.aspx 文件中，其中编程逻辑以代码声明块的形式嵌入到 script 中，放在程序的前面。而由 HTML 标记、静态文本和 ASP.NET 服务器控件构成的可视元素则放在程序的后面，如例 6-1。

（2）代码隐藏页模式，是 ASP.NET 新引入的一种代码绑定技术，它将可视元素和编程逻辑分别放置在两个文件中。其中实现界面设计的可视元素仍存放在扩展名为.aspx 的文件中，而由服务器执行的编程逻辑则存放在扩展名为.aspx.cs（假设此处使用的程序设计语言是 C#）的文件中。为了实现两个文件的关联，必须对.aspx 文件中 Page 指令的 CodeBehind 属性进行设置。若.aspx 文件名为 login.aspx，则 CodeBehind 属性应设置为：

```
<%@PageLanguage="C#" AutoEventWireup="false" CodeBehind="login.aspx.cs" Inherits="login" %>
```

这一模式对于代码的重用、程序的调试和维护均有重要意义。采用代码隐藏页模式还可以有效地保护代码，提高程序的安全性。

若例 6-1 采用代码隐藏页模式，代码将分别存放在两个文件中。其中 login.aspx 文件的内容如下：

```
<%@ Page Language="C#" AutoEventWireup="true" CodeBehind="login.aspx.cs" Inherits="login" %>
<html>
<head runat="server">
    <title>无标题页</title>
</head>
<body>
    <form id="form1" runat="server">
    用户登录
    <hr />
<%--下面构造一个表单--%>
    <div>
            <!--RadioButton 单选钮  -->
            <asp:RadioButton ID="Radiostudent" runat="server" checked="true"
            GroupName="sel" Text="学生" />
        <asp:RadioButton ID="Radioteacher" runat="server" GroupName="sel" Text="教师" />
        <br />
         <!--TextBox 单行文本输入框-->
         用户名：<asp:TextBox ID="usename" runat="server"></asp:TextBox>
        <br />
          <!--TextBox 密码文本输入框-->
         密码：
        <asp:TextBox ID="usepassword" runat="server" TextMode="Password"></asp:TextBox>
        <br />
        <asp:Button ID="Button1" runat="server"    onclick="Button1_Click" Text="登录" />
        <asp:Button ID="Button2" runat="server" onclick="Button2_Click" Text="取消" />
     </div>
    <!--输出当前日期和时间-->
     登录时间是：<%=DateTime.Now%>
    </form>
</body>
</html>
```

login.aspx.cs 文件的内容如下：

```
using System;
using System.Collections;
using System.Configuration;
using System.Data;
using System.Linq;
using System.Web;
using System.Web.Security;
using System.Web.UI;
using System.Web.UI.HtmlControls;
using System.Web.UI.WebControls;
using System.Web.UI.WebControls.WebParts;
using System.Xml.Linq;
public partial class login : System.Web.UI.Page
{       //登录按钮的单击事件过程
    protected void Button1_Click(object sender, EventArgs e)
        {
            if (usename.Text == "administrator" && Radioteacher.Checked == true
&& usepassword.Text != "")
```

```
                    //输出身份是管理员的欢迎词
                    Response.Write("欢迎你管理员同志！");
                else if (Radioteacher.Checked == true && usepassword.Text != "")
                    Response.Write("欢迎你" + usename.Text + "老师！"); //身份是老师
                else if (usepassword.Text != "")
                    Response.Write("欢迎你" + usename.Text + "同学！"); //身份是学生

            }
        protected void Button2_Click(object sender, EventArgs e)
            {
                usename.Text = "";
                usepassword.Text = "";
                usename.Focus();
            }
    }
```

3．ASP.NET 的文件类型

一个完整的 ASP.NET 应用，指某个虚拟目录及其子目录中 Web 服务、Web 页面、服务器控件、执行代码及配置参数等所有文件的综合。ASP.NET 的文件用不同类型的扩展名加以区分，下面是 ASP.NET 中几种常用的文件类型。

（1）aspx 页面文件。该文件由可视元素和编程逻辑两部分组成，如同过去的.asp 文件，浏览器可执行此类文件，向服务器提出浏览请求。

（2）ascx 用户控件文件。内含用户控件的文件，可包含在多个.aspx 文件中。

（3）resx 资源文件。资源是在逻辑上由应用程序部署的任何非可执行数据。通过在资源文件中存储数据，无须重新编译整个应用程序即可更改数据。

（4）aspx.cs 或.aspx.vb 代码分离文件。.aspx.cs，用 C#语言编写的.aspx 页面的后台代码文件。.aspx.vb 用 VB.NET 语言编写的.aspx 页面的后台代码文件。

（5）ascx.cs 或.ascx.vb 文件。用户控件的代码分离文件。

（6）sln 解决方案文件。为 Visual Studio.NET 提供对项目、解决方案项的引用。

（7）Web.config 配置文件。该文件向它所在的目录和所有子目录提供配置信息。

（8）Global.asax 配置文件。ASP.NET 系统环境设置文件，相当于 ASP 中的 global.asa 文件。

（9）Master 母版页文件。该文件为应用程序中的所有页（或一组页）定义统一的外观和行为。

4．ASP.NET 事件驱动的编程模型

ASP.NET 采用事件驱动的编程模型，添加到网页上的 Web 服务器控件通过所触发的事件来执行系列操作。但 ASP.NET 事件的概念与传统的基于客户端程序的事件概念不同，主要是事件的触发位置和事件的处理位置不同。基于客户端程序的事件在客户端触发，在客户端处理，而 ASP.NET 的事件不论发生在客户端还是发生在服务器端基本上都是在服务器端处理。在客户端触发的事件要求客户端提供一个俘获事件信息并将该事件信息传递到服务器上的机制。服务器接收到该信息后，ASP.NET 解析该信息，找出事件的类型，查找并调用相应的事件处理过程。从俘获信息、传输、解析、控制权转移到事件处理过程，都是由 ASP.NET 的服务器负责，无须人工干预。但是，服务器端事件的响应过程是一个服务器和客户端交互的过程，由于受网络带宽的限制，如果设计过多的服务器端响应事件，会造成事件信息在客户端和服务器端频繁传递，从而降低网络的性能。为此，在服务器端响应的事件是十分有限的。为了减少事件信息的频繁传递，客户端发生的事件，并不是每发生一次就向服务器传送一次信息。默认情况下，只有当服务器控件按钮（Button）被单

击时，才向服务器传递事件信息。其他控件的事件，一般都是先保存在客户端的缓冲区，等到下一次向服务器传递信息时，才和其他信息一起传送到服务器。如需要立即得到响应，只需将该控件的 AutoPostBack 属性设置为 true。

ASP.NET 中的事件有：HTML 事件，在 ASP.NET 生成页面时自动触发的几个事件以及用户与页面交互时触发的事件等。ASP.NET 页面事件处理过程如图 6-2 所示。

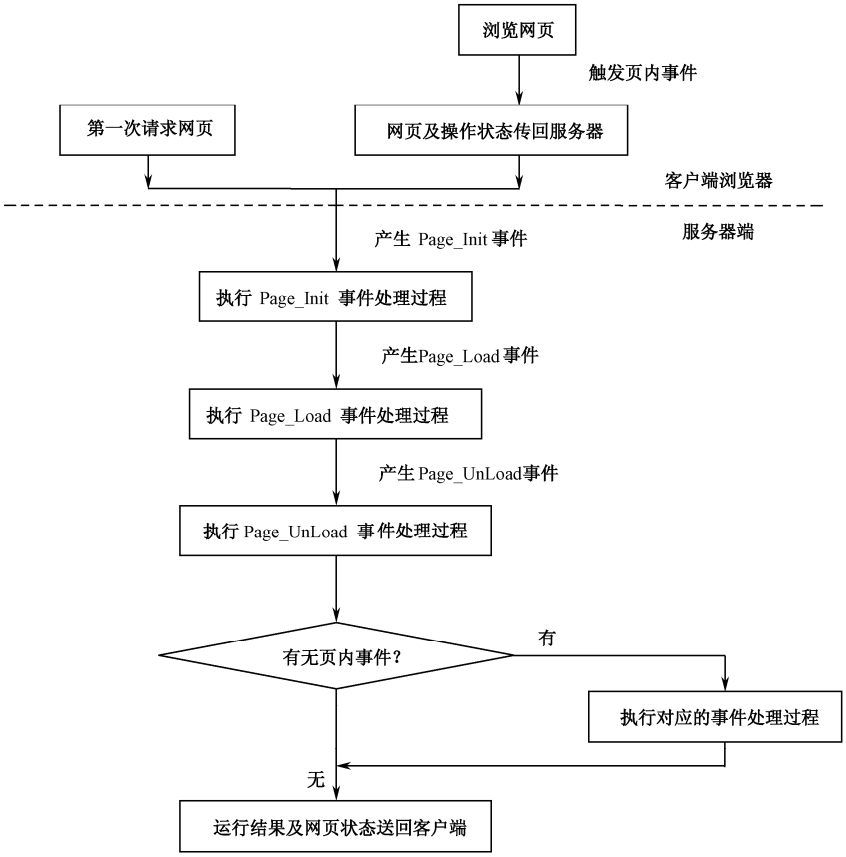


图 6-2　ASP.NET 页面事件处理过程

6.1.3　命名空间

命名空间又称名称空间或名字空间。它将一些提供相似功能或具有相似状态的类聚合在一起组成一个在逻辑上相关的单元，以便在.NET 中使用，它是.NET 框架的重要组成部分。命名空间采用树状结构的管理方式，每一层之间用“ . ”隔开，记录类的名称及其所在的位置。命名空间不仅由类和对象组成，而且含有子命名空间，如 System.data.sqlclient 就是 System.data 的子命名空间。在.NET 系统类库中包含 80 多个命名空间，命名空间 System.IO 的一个实例就包含了那些用于处理输入和输出操作的类。

为了在 ASP.NET 页面中使用这些类，必须使用前面介绍的页面指令@Import 将命名空间导入到 aspx 页面。若将命名空间导入到扩展名为.aspx.cs 的后台页面中，则需使用关键字 using。例如，将以下命名空间导入到 aspx 页面中使用下述页面指令：

```
<%@ Import Namespace="System.data"%>
<%@ Import Namespace="System.data.sqlclient"%>
<%@ Import Namespace=" System.IO" %>
```

将以上命名空间导入到.aspx.cs 后台页面，使用以下方法：

```
using System.data;
using System.data.sqlclient;
using System.IO;
```

实际上，命名空间 System.IO 无须显式导入。默认情况下，下列命名空间会自动导入到每一个 ASP.NET 页面中。

System：包含所有基本数据类型和其他诸如与生成随机数、处理日期和时间相关的那些类。

System.Colletions：包含处理诸如哈希表（散列表）和数组列表等标准集合类型的类。

System.Collections.Specialized：包含表示链表和字符串集合等特定集合的类。

System.Configuration：包含处理配置文件的类。

System.IO：包含读/写数据流/文档和普通输入/输出（I/O）功能的类型和类。

System.Text：包含编码、解码和操作字符串内容的类。

System.Text.RegularExpressions：包含执行正则表达式匹配和替换操作的类。

System.Web：包含使用万维网的基本类，其中包括表示浏览器请求和服务器响应的类。

System.Web.Caching：包含缓存页面内容和执行自定义缓存操作的类。

System.Web.Security：包含实现验证和授权的类。

System.Web.SessionSate：包含实现会话状态的类。

System.Web.UI：包含构建用户界面的基本类。

System.Web.UI.HtmlControls：包含 HTML 控件的类。

System.Web.UI.WebControls：包含 Web 控件的类。

6.2 C#语言基础

ASP.NET 运行在公共语言运行库之上，所有符合通用语言规范的编程语言都可以用来编写 Web 应用程序，ASP.NET 支持多种编程语言，如 Visual Basic.NET、C# 和 JScript .NET 等。其中 C#语言是 Microsoft 公司专门为.NET 平台精心设计和量身定制的程序设计语言，它不仅保留了 Java 语言的简洁性和 Visual Basic 语言的易用性，而且继承了 C++语言的面向对象的特性，既可以作为 Windows 应用程序，又可以作为 Web 应用程序的编程语言。C#的语法结构简单、功能强大，与 C 和 C++语言极为相似，如果读者熟悉 C 和 C++语言，就可以快速掌握 C#。

6.2.1 C#语法规则

1．一个简单的 C#程序

【例 6-2】用 C#语言编写一个控制台应用程序。

```
using System;
class Program
{
        // A "Hello World!" program in C#
        static void Main()
        {
            Console.WriteLine("Hello World!"); //控制台输出
            Console.ReadLine(); //等待控制台输入
        }
}
```

运行上述程序，屏幕显示“ Hello World! ”。

2．C#程序结构和编写规则

（1）用 using 指令导入需要的命名空间。

（2）用 class 定义一个类。

（3）每个 C#程序有且仅有一个 Main()方法，用于控制程序的开始和结束。程序的执行总是从 Main()方法开始。

（4）大括号“ { ”和“ } ”表示程序中某个代码块的开始和结束，左括号和右括号必须配对使用。

（5）每一条语句都以分号“ ; ”结尾。多个语句可以写在一行上。例如：

```
Temp=a;a=b;b=temp;      //交换变量 a,b 的值，一行写三个语句
```

（6）C#语句区分大小写，NO、No 被认为是不同的。

（7）C#的注释分为行注释和块注释。行注释用“ // ”表示。块注释以“ /* ”开头，以“ */ ”结尾。

6.2.2　数据类型与变量

1．数据类型

数据类型决定了存储数据和处理数据的方式。在 C#中，数据类型被分为值类型和引用类型两大类。两者的区别在于值类型的变量直接存放实际的数据，可以直接访问。而引用类型的变量存放的是数据的地址，不可直接访问。对于引用类型，两个变量可能引用同一个对象，因此对一个变量的操作可能影响另一个变量所引用的对象。对于值类型，每个变量都有自己的数据副本，对一个变量的操作不会影响到另一个变量。C#语言的数据类型分类如图 6-3 所示。

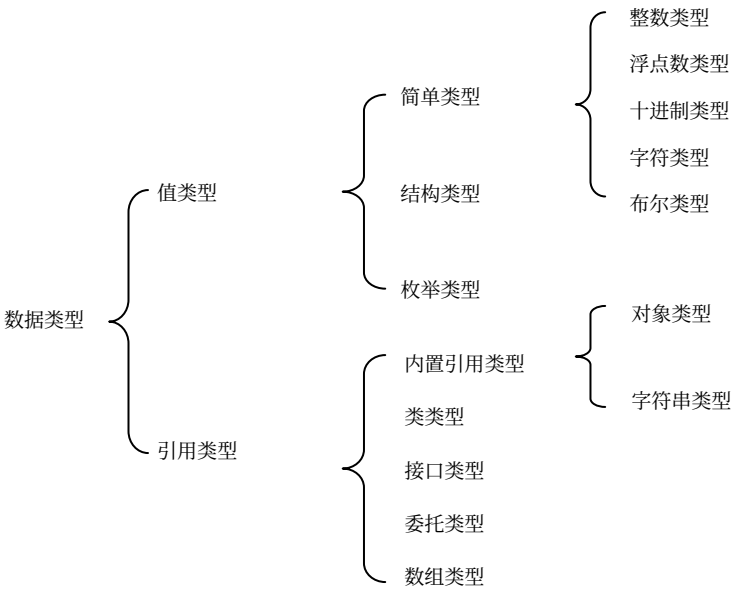


图 6-3　C#数据类型分类

（1）值类型

值类型包括简单类型、结构类型和枚举类型。其中简单类型又进一步细分为：整数类型、字符类型、浮点数类型、十进制类型和布尔类型。

① 简单类型

C#简单类型及其取值范围如表 6-2 所示。

表 6-2　C#简单类型及其取值范围

保　留　字	System 命名空间中的名字	字　节　数	取 值 范 围
sbyte	System.Sbyte	1	-128~127
byte	System.Byte	1	0~255
short	System.Int16	2	-32768~32767
ushort	System.UInt16	2	0~65535
int	System.Int32	4	-2147483648~2147483647
uint	System.UInt32	4	0~4294967295
long	System.Int64	8	-9223372036854775808~9223372036854775807
ulong	System.UInt64	8	0~18446744073709551615
char	System.Char	2	一个 Unicode 字符
float	System.Single	4	-3.402823e38~3.402823e38
double	System.Double	8	-1.79769313486232e308~1.79769313486232e308
decimal	System.Decimal	16	-79228162514264337593543950335~79228162514264337593543950335
bool	System.Boolean	4	true 或 false

C#简单类型使用方法和 C、C++中相应的数据类型基本一致。需要注意如下几点。

- C#每种数据类型所占字节数是一定的。
- 字符类型采用 Unicode 字符集，一个 Unicode 标准字符长度为 16 位。字符型数据用单引号括起。可以按以下方法直接给一个字符型变量赋值：char c='A'。也可以通过十六进制转义字符（前缀“ \x ”加十六进制数字）或 Unicode 转义符（前缀“ \u ”加十六进制数字）给字符变量赋值。例如：char c='\x0032'和 char c='\u0032'均表示将字符“ 2 ”赋值给字符变量 c。同 C 和 C++一样，C#也可以使用转义字符，通过“ \ ”加其他字符来表示。C#中的转义字符如表 6-3 所示。

表 6-3　C#转义字符

转　义　符	字　符　名	转　义　符	字　符　名
\’	单引号	\”	双引号
\\	反斜杠	\0	空字符
\a	警报	\b	退格符
\f	换页符	\n	换行符
\r	回车	\t	水平制表符
\v	垂直制表符		

- 布尔类型有两个值：false 和 true。在 C#中，不能认为整数 0 是 false，其它值是 true。bool x=1 是错误的，不存在这种写法，只能写成 bool x=true 或 bool x=false。
- 一般带小数点的数或用科学计数法表示的数都被认为是浮点数。浮点数的数据类型默认为 double 类型，可以通过加后缀的方式改变其默认类型。例如：

```
double f1=3.14,f2=12.34e5;          //默认 3.14，12.34e5 为 double 型
float f3=45.78f;                    //加字符“ f ”，45.78 为 float 型
double f4=45.78e-5d;                //加字符“ d ”，45.78e-5 为 double 型
```

② 结构类型

结构类型是用户自定义的数据类型，它是多个相关联的不同类型的数据组合在一起形成的类

型。既包含数据成员，也包含对数据操作的函数成员。结构类型必须先定义后使用。结构类型的定义格式如下：

```
struct  结构类型名
{
    结构成员定义;
}
```

例如：下面定义了一个表示学生成绩信息的结构类型 Grade，包含了学号、数学、物理、均分 4 个不同类型的数据信息。

```
struct Grade
{
    public long no;
    public int math, physics;
    public double ave;
}
```

使用该结构类型，代码如下：

```
Grade s1;   //声明了一个结构变量 s1
s1.no = 10001; //给变量成员 no 赋值
s1.math = 99; //给变量成员 math 赋值
s1.physics = 97; //给变量成员 physics 赋值
s1.ave = (s1.math + s1.physics) / 2.0; //将计算结果赋值给变量成员 ave
```

③ 枚举类型

枚举类型也是一种自定义的数据类型，它指定了该数据类型变量可以拥有的所有合法值，并且允许用符号代表数据。枚举类型的定义格式为：

```
enum  枚举类型名 {枚举常量列表}
```

例如：

```
enum weekday {sunday, monday, tuesday, wednesday, thursday, friday, saturday}
```

在枚举类型中，每个元素的默认类型为 int，第一个元素的值默认为 0。因而，枚举类型 weekday 中的 sunday 元素值为 0，monday 元素值为 1，tuesday 元素值为 2，依此类推。当然，也可以给枚举元素直接指定值。例如：

```
enum weekday {sunday=7,monday=1, tuesday, wednesday, thursday, friday, saturday}
```

指定 sunday 值为 7，monday 值为 1，以后按顺序依次加 1，saturday 值为 6。

使用枚举类型 weekday，代码如下：

```
weekday day1=weekday.tuesday; //声明枚举变量 day1，初始化值为 tuesday
weekday day2；//声明枚举变量 day2
day2=(weekday)7; //给枚举变量 day2 赋值 sunday
```

（2）引用类型

引用类型包括对象类型、字符串类型、数组类型、类类型、接口类型和委托类型。其中对象类型和字符串类型属于内置引用类型。

① 对象类型

对象类是所有其他类的基类，C#中的每种类型都是直接或间接从对象类派生的。因此，可以将任何类型的值赋给对象类型的变量。例如：

```
int x=25;
object obj1;
obj1=x;
object obj2= 'A';
```

其中关键字 object 是在命名空间 System 中定义的，是类 System.Object 的别名。例如：

```
int i=123;
```

```
object box=i;       //装箱
int j = (int)box;     //拆箱
```

上述将值类型的变量 i 转换为对象的过程称为“ 装箱 ”，将对象类型的变量 box 转换为值类型的过程称为“ 拆箱 ”。装箱和拆箱的概念是 C#的类型系统的核心，它在值类型和引用类型之间架起了一座桥梁，使得任何值类型的值都可以转换为对象类型的值，反之亦可。

② 字符串（string）类型

字符串类型直接从对象类中继承而来，表示一串字符序列。字符串型数据用一对英文双引号括起来。例如：

```
string s="欢迎学习 C#!";
//声明字符串变量 s，初始化值为字符串"欢迎学习 C#!"
```

其中关键字 string 是类 System.String 的别名。

注意：“ A ”与‘A’有着本质区别，前者是字符串型数据，后者是字符型数据。

③ 数组类型

数组是同类型变量的有序集合，集合中的变量称为数组元素。数组中所有元素具有相同的数据类型和名称，并依据各自不同的下标值相互区分。数组的下标从 0 开始，即第一个元素的下标是 0，以后元素依次递增。只有一个下标的数组称为一维数组，有多个下标的数组称为多维数组。数组元素的类型可以为任意类型，也可以是数组类型，即数组的元素还是数组。数组必须先声明后使用。例如：

```
int [] c   ; //声明了一个任意大小的一维数组 c
int [] a=new int[3] ; //声明了一个由 3 个整型元素组成的一维数组 a
a[0]=4 ;a[1]=5 ;a[2]=6 ; //给数组 a 元素赋值
int [,] d ; //声明了一个任意大小的二维数组 d
int [,] b =new int[2,3];
//声明了一个 2 行 3 列的二维数组 b，共计有 b[0,0],b[1,0],b[0,1],b[1,1],b[0,2],b[1,2]6 个元素。
int [][] c ;//声明一个数组的数组
```

在声明数组的时候可以直接将其初始化。例如：

```
string [] c=new string[]{"one","two","three"};
```

或

```
string [] c={"one","two","three"};
```

数组元素 c[0]，c[1]，c[2]的值分别为字符串“ one ”，“ two ”,“ three ”。

再例：

```
 int [,] b=new int [,]{{1,2,3,},{4,5,6}};
```

声明了一个含有 6 个元素的二维数组，元素 b[0,0]，b[0,1]，b[0,2]，b[1,0]，b[1,1]，b[1,2]的值分别为 1，2，3，4，5，6。

④ 类类型

类是一组具有相同数据结构和相同操作的对象集合，是 C#中功能最为强大的数据类型。它和结构的不同在于：类是引用类型，结构是值类型，并且类支持继承。有关类的详细介绍见 6.2.6。

⑤ 接口类型

接口就是一个约定，它与类极为相似，可以有属性、事件和方法。但接口只定义成员的名称，并不提供实现的代码，只能在具体的类或结构代码中实现。接口不能实例化，只能从接口派生类，从某接口派生的类必须遵守该接口定义的约定。通过接口可以实现多重继承。

⑥ 委托类型

委托顾名思义就是中间代理人，类似于其他语言中的函数指针的概念，但是与函数指针不同，委托是面向对象的，并且类型安全可靠。通过委托，我们能够将方法作为实体赋值给变量或作为参数传递。一旦为委托分配了方法，委托将与该方法具有完全相同的行为。委托实际上就是对与

之签名（返回类型和参数）相一致的类或结构方法的引用，并且委托可以引用静态方法和实例方法。

2．常量和变量

常量和变量是程序设计中经常会涉及到的一个重要概念。它们通常代表程序中的数据。

（1）常量

程序执行过程中其值始终不变的量。C#中有两种类型的常量：直接常量和符号常量。

① 直接常量

在程序中直接给出的数据值。例如：12、3.14、23.12e5、5.15f、"欢迎学习 C#！"、true、'A'等。

② 符号常量

用 const 语句声明的用户自定义常量，一旦赋值其值就不可改变。常量定义的语法格式如下：

```
const 类型名  常量名=常量表达式；
```

例如：

```
const double PI =3.1415926;
const double AREA = PI*2*2;
```

若再有语句：AREA= PI*3*3;程序报错。

（2）变量

程序执行过程中其值可以变化的量。每一个变量都有一个名称和数据类型。

① 变量名的命名规则

变量名由 26 个英文字母、数字、下画线组成，且必须以字母或下画线开头。变量名中间不能有空格。不能与 C#中的关键字、库函数同名。

② 变量的声明

C#是一种强类型语言，所以变量必须“ 先声明，后使用 ”。声明变量就是通知应用程序按照变量的类型事先为其分配适当的存储空间。变量声明语句的语法格式如下：

```
数据类型名  变量名 1，变量名 2，变量名 3，……，变量名 n；
```

例如：

```
int x，y;
bool z=true;
```

声明了两个整型变量 x、y 和一个逻辑型变量 z。

③ 变量的初始化和赋值

C#语言允许声明变量的同时给变量赋一个初值。例如：

```
double a, b=13.67;
char c1='y',c2='n';
string str1 = "Hello";
```

也可以先声明变量，后初始化赋值。例如：

```
float d;       //在声明变量时未初始化变量 d
d=33.65f;     //用赋值语句给变量 d 初始化赋值
```

在 C#中不允许引用未初始化的变量。

④ 变量的作用域

变量的作用域即变量的作用范围，分为局部作用域和类作用域。

- 局部作用域

在某个区域块中定义的变量，只能在这个区域块中使用。例如，在方法中声明的变量就具有那个方法的作用域，从界定方法主体的大括号“ { ”起始到“ } ”结束。一旦方法结束，它们也会消失，而且只能由那个方法内部执行的代码来访问。具有此作用域的变量称为局部变量。

- 类作用域

在类定义体中定义的变量，具有那个类的作用域，从界定类主体的大括号“{”起始到“}”结束。在 C#中，我们把具有此作用域的变量称为字段。和局部变量不同，使用字段可以在不同的方法之间共享信息。例如：

```
class Program
{
        static int a=2; //具有类作用域的变量 a（字段）
        public static void B()
        {
            int b= 1;//局部变量 b，仅作用在方法 B 中
         }
        public static void C()
        {
            int c= 0;//局部变量 c，仅作用在方法 C 中
            c=a; //引用变量 a 的值
            c=b; //此句出错！引用局部变量 b，超出了方法 B 的作用域
         }
 }
```

在 Program 这个类里，a 是具有类作用域的变量，变量 b 和变量 c 都是局部变量。在方法 C 中，可以访问变量 a，但不可访问局部变量 b。因为变量 b 的作用域只是在方法 B 的“{”、“}”之间，而变量 a 的作用域在类 Program 的“{”、“}”之间。

3．类型转换

C#语言是一种强类型语言，它的数据类型有一些可以进行隐式转换，其他的必须进行强制类型转换。转换规则如下。

（1）隐式转换只能是长度短的类型向长度长的类型转换。如 int 可以转换成 long、float、double，反之必须强制转换。例如：int a=20;float b=a;正确。再例如：float b=12.34f;int a=b;不正确。

（2）十进制类型(decimal)也是浮点数类型，具有较高的精度，但取值范围比较小，浮点类型和十进制类型之间的转换可能会产生溢出异常或精度损失，所以它们之间不存在隐式转换，只能是强制类型转换。例如：

```
decimal a = 12.356799543m;
float b;
b = (float)a;
```

结果变量 b 的值为 12.3568。

（3）整数类型不能隐式转换为字符类型(char)。例如：char c=65 是错误的，必须写成 char c=(char)65。

（4）值类型可以隐式转换为引用类型，例如：object box = 123。反之，必须是强制转换，例如：int j = (int)box。

6.2.3 运算符与表达式

由常量、变量、对象及各种运算符组成的式子称为表达式。表达式中的常量、变量、对象统称为操作数。运算符是执行某种运算功能的符号，用于执行计算，为变量赋值，进行相等或不相等测试以及其他一些操作。C#提供了大量的运算符，常用的有算术运算符、赋值运算符、比较运算符、逻辑运算符、字符串连接运算符、typeof 运算符、new 运算符等。

1．算术运算符与算术表达式

算术运算符就是进行数学计算的运算符。C#算术运算符如表 6-4 所示。

表 6-4　C#算术运算符

运算符分类	运　算　符	说　　明	举　　例	运 算 结 果
一元运算符	+	取正	+9	9
	-	取负（负号）	-3	-3
	++	操作数加 1	i=3; j=i++; i=3; j=++i	i 的初值为 3，j 的值为 3 i 的初值为 3，j 的值为 4
	--	操作数减 1	i=3; j=i--; i=3; j=--i	i 的初值为 3，j 的值为 3 i 的初值为 3，j 的值为 2
二元运算符	+	加	4+5	9
	-	减	10-8	2
	*	乘	4*2	8
	/	除	7/2 7.0/2	3 3.5
	%	求余	7%5	2

2．赋值运算符与赋值表达式

赋值运算符为数值、枚举、类等类型变量赋值。C#赋值运算符如表 6-5 所示。

表 6-5　C#赋值运算符

运　算　符	说　　明	举　　例	运 算 结 果
=	给变量赋值	int a,b;a=1;b=a	运算后 b 的值为 1
+=	操作数 1 与操作数 2 相加后赋值给操作数 1	int a,b; a=2;b=3; b+=a	b+=a 等价于 b=b+a，运算后 b 的值为 5
-=	操作数 1 与操作数 2 相减后赋值给操作数 1	int a,b; a=2;b=3; b-=a	b-=a 等价于 b=b-a，运算后 b 的值为 1
*=	操作数 1 与操作数 2 相乘后赋值给操作数 1	int a,b; a=2;b=3; b*=a	b*=a 等价于 b=b*a，运算后 b 的值为 6
/=	操作数 1 与操作数 2 相除后赋值给操作数 1	int a,b; a=2;b=6; b/=a	b/=a 等价于 b=b/a，运算后 b 的值为 3
%=	操作数 1 与操作数 2 相除取余赋值给操作数 1	int a,b; a=2;b=6; b%=a	b%=a 等价于 b=b%a，运算后 b 的值为 0

3．比较运算符与比较表达式

用于两个值的比较，结果是一个布尔值，即真（true）或假（false）。常用的比较运算符如表 6-6 所示。

表 6-6　C#比较运算符

运　算　符	说　　明	举　　例	运 算 结 果
>	大于	'a'>'A'	true
>=	大于等于	3.41>=3.4	true
<	小于	27<7	false
<=	小于等于	'a'<='b'	true
==	等于	"ABC"=="ABBC"	false
!=	不等于	"a"!="A"	true
is	检查表达式是否为指定类型	1 is float 1.0d is double	false true

（1）两个字符数据比较大小时，比较的是字符 ASCII 码值的大小。

（2）只有运算符“ == ”和“ != ”才可用于字符串比较。例如，有如下语句：

```
string a = "hello";
string b = "h";
b += "ello";
```

那么 a == b 的结果为 true, (object)a == (object)b 的结果为 false。这是因为相等运算符(== 和 !=)被定义为比较 string 对象的“ 值 ”，此处字符串的内容相同，所以值为 true。但是 a 和 b 引用的不是同一个字符串实例，所以结果为 false。若将上述语句改为：

```
string a = "hello";
string b = a;
```

此时 a == b 的结果为 true，(object)a == (object)b 的结果也为 true。

（3）Is 运算符的使用格式为：e is T，其中 e 是一个表达式，T 是一个类型，该式判断 e 是否为 T 类型，一致结果为 true，否则为 false。

4．逻辑运算符与逻辑表达式

用于执行逻辑运算。参加运算的数据必须为布尔型数据，结果亦为布尔型。常用的逻辑运算符如表 6-7 所示。

表 6-7　C#逻辑运算符

运　算　符	说　　明	举　　例	运 算 结 果
!	非运算。取反操作	!false !true	true false
&&	与运算。只有两个操作数均为 true 时，结果才为 true	true && false true && true	false true
||	或运算。只要有一个操作数为 true，结果就为 true	true || false false || false	true false

5．条件运算符与条件表达式

条件运算符“ ?: ”是唯一的一个三元运算符。对应的条件表达式的格式如下：

```
e ? x:y;
```

如果表达式 e 的值为 true，则计算表达式 x，x 的值就是条件表达式的值；如果 e 的值为 false，则计算表达式 y，y 的值就是条件表达式的值。例如：

```
b=a>=0? 2*3:4-5;
```

如果 a 的值大于等于 0，则 b 的值为 6，否则为-1。

6．字符串连接运算符

将两个字符串拼接成一个字符串，使用“ + ”运算符。例如：

```
string str;
str="欢迎学习"+"C#！";
```

变量 str 的值为：欢迎学习 C#!

7．typeof 运算符

用于获得指定类型在 system 名称空间中定义的类型名字。使用格式：

```
typeof(类型名);
```

例如：

```
typeof(int) ;                值为：System.Int32
typeof(System.Int32) ;       值为：System.Int32
typeof(string) ;             值为：System.String
```

由此可见，int 和 System.Int32 是同一类型。

8．new 运算符

用于创建值类型、类类型、数组类型和委托类型的实例。例如：

```
int x=new int();//用 new 创建整型变量 x
Student c1=new Student ();
//用 new 创建 Student 类对象，变量 c1 是 Student 类对象的引用。
int[] arr=new int[2];//数组也是类，创建数组类对象，arr 是数组对象的引用
```

9．运算符的优先级及结合性

表达式中各种运算符的计算顺序由运算符的优先级和结合性决定。C#运算符的优先级和结合性如表 6-8 所示。

表 6-8　C#运算符从高到低的优先级和结合性

类　别	操　作　符	结　合　性
初级操作符	(x) x.y f(x) a[x] x++ x-- new typeof sizeof checked unchecked	
一元操作符	+ - ! ~ ++x--x (T)x	自右向左
乘除操作符	* / %	自左向右
加减操作符	+ -	自左向右
移位操作符	<< >>	自左向右
关系操作符	< > <= >= is as	自左向右
等式操作符	== !=	自左向右
逻辑与操作符	&	自左向右
逻辑异或操作符	^	自左向右
逻辑或操作符	|	自左向右
条件与操作符	&&	自左向右
条件或操作符	||	自左向右
条件操作符	?:	自右向左
赋值操作符	= *= /= %= += -= <<= >>= &= ^= |=	自右向左

（1）当一个表达式包含多种操作符时，操作符的优先级控制着操作符求值的顺序。高优先级的先参加运算，低优先级的后参加运算。

（2）当表达式中出现两个相同优先级的运算符时，根据它们的结合性进行计算。左结合运算符按从左到右的顺序计算。例如，x* y / z 计算为 (x * y) / z。右结合运算符按从右到左的顺序计

算。例如，x = y = z 按 x = (y = z) 计算。赋值运算符和三元运算符 (?:) 是右结合运算符，其他所有二元运算符都是左结合运算符。

（3）优先级和结合性顺序可以用括号改变。例如，x + y * z 先将 y 乘以 z，然后将结果与 x 相加，而(x + y) * z 先将 x 与 y 相加，然后再将结果乘以 z。

6.2.4 流程控制语句

任何程序都是由语句组成的，C#为我们提供了实现多种功能的语句。如前面已经介绍过的常量定义语句、变量声明语句和求解表达式值的表达式语句。此处的表达式包括方法调用、使用 new 运算符的对象分配、使用“ = ”和复合赋值运算符的赋值，以及使用“ ++ ”和“ -- ”运算符的增量和减量运算。一般情况下，程序按照语句的书写顺序依次执行，这种程序结构称之为顺序结构。下面将给大家介绍的是可以改变程序执行顺序的流程控制语句。

1．选择语句

根据条件表达式的值从若干个给定的语句中选择一个来执行。这组语句有 if 和 switch 语句。

（1）if 语句

if 语句是最常用的选择语句，有三种格式。

① 格式 1：

```
if （条件表达式）{语句组;}
```

当条件表达式为真（true，条件成立）时，执行语句组，否则什么也不做。“ {} ”括起来的语句组可以是多条语句，也可以是一条语句。当只有一条语句时，“ {} ”省略。

② 格式 2：

```
if （条件表达式）{语句组 1;}
else {语句组 2;}
```

当条件表达式为真（true，条件成立）时，执行语句组 1，否则（false，条件不成立），执行语句组 2。

③格式 3：

```
if （条件表达式 1）{语句组 1;}
elseif （条件表达式 2）{语句组 2;}
…
elseif （条件表达式 n）{语句组 n;}
else {语句组 n+1;}
```

如果条件表达式 1 的值为 true，执行语句组 1；否则，如果条件表达式 2 的值为 true，执行语句组 2；……，如果前面所有条件表达式都不成立，则执行语句组 n+1。

【例 6-3】根据学生的分数，评定不同的成绩等级，其中分数 score 值由随机函数产生。分数 90 及以上评定为优秀，分数 80～89 之间评定为良好，分数 70～79 之间评定为中，分数 60～69 之间评定为及格，60 以下为不及格。

```
using System;
class Program
{
    static void Main()
    {
       Random rad = new Random();//实例化随机数产生器 rad；
       int score;   //分数
       string grade;//等级
       score= rad.Next(1, 100);//用 rad 生成大于等于 1，小于等于 100 的随机数
```

```
            if (score>=90)
                grade="优秀";
            else if(score>=80)
                grade="良好";
            else if(score>=70)
                grade="中";
            else if(score>=60)
                grade="及格";
            else
                grade="不及格";
            Console.WriteLine("等级：" + grade);
            }
    }
```

（2）switch 语句

格式：

```
    switch (表达式)
    {
    case  常量表达式 1:语句组 1;break;
    case  常量表达式 2:语句组 2;break;
    ...
    case  常量表达式 n:语句组 n;break;
    [default:语句组 n+1;break;]
    }
```

switch 语句的执行过程是：先求“ 表达式 ”的值，然后将该值与 case 子句中的“ 常量表达式 ”依次进行比较。若与某个“ 常量表达式 ”相匹配，执行该 case 子句中的语句组，随后执行 break 语句，结束 switch 结构。若与所有“ 常量表达式 ”均不匹配，则执行 default 子句中的语句组 n+1。

【例 6-4】同例 6-3，用 switch 语句实现。

```
    using System;
    class Program
    {
        static void Main()
        {
            int score;   //分数
            string grade;//等级
            Random rad = new Random();//实例化随机数产生器 rad；
            score = rad.Next(1, 100);//用 rad 生成大于等于 1，小于等于 100 的随机数
            switch (score / 10)
            {
                case 10:
                case 9: grade = "优秀"; break;
                case 8: grade = "良好"; break;
                case 7: grade = "中"; break;
                case 6: grade = "及格"; break;
                default: grade = "不及格"; break;
            }
          Console.WriteLine("等级：" + grade);
        }
    }
```

使用 switch 语句必须注意以下几点。

①　“ 常量表达式 ”值的数据类型须与“ 表达式 ”值的类型相一致或者能够隐式转换为“ 表达式 ”类型。

② break 语句是可选项，用来终止 switch 语句。任何时候遇到 break 语句都会结束 switch 语句的运行。

③ default 语句也是可选项，是当各个“常量表达式”的值均不匹配“表达式”值时的选择。

2．循环语句

用于有规律地重复执行某一程序块。C#提供了四种循环语句： do－while 循环语句、while 循环语句、for 循环语句和 foreach 循环语句。

（1）do－while 循环语句

格式：

```
do {语句组;} while (条件表达式）;
```

首先执行语句组，然后计算条件表达式的值。若条件表达式值为 true，继续下一次循环；若条件表达式的值为 false，结束循环。例如：

```
int n = 1,fact=1;
do {
    fact *= n;
    n++;
}
while (n < 6);
```

该段程序的功能是求 5!，循环结束后 fact 的值为 120。

注意：while()后面的分号“;”不能缺少。

（2）while 循环语句

格式：

```
while（条件表达式）{语句组;}
```

首先计算条件表达式的值，若条件表达式值为 true，执行语句组，若条件表达式的值为 false，结束循环。

do－while 循环与 while 循环区别在于：do－while 循环语句中的语句组至少执行一次，while 循环的语句组可能一次都不执行。

（3）for 循环语句

格式：

```
for ([表达式 1];[表达式 2];[表达式 3]) {语句组;}
```

其中，表达式 1，表达式 2，表达式 3 都是可选项。for 循环的执行过程如下。

① 计算表达式 1 的值。

② 计算表达式 2 的值。如果表达式 2 的值为 false，转向步骤④；如果表达式 2 的值为 true，则执行语句组，接着计算表达式 3 的值。

③ 转向步骤②。

④ 结束循环，执行 for 循环后的下一条语句。

（4）foreach 循环语句

格式：

```
foreach(数据类型名 变量 in 集合对象) {语句组;}
```

该语句用于遍历集合中的各个元素。执行过程中，“变量”代表“集合对象”中的每一个元素。每循环一次，取“集合对象”中的元素一次。

【例 6-5】用 foreach 语句输出数组的值。

```
using System;
class Program
```

```
{
    static void Main()
    {
        double[,] values = { { 1.2, 2.3, 3.4, 4.5 }, { 5.6, 6.7, 7.8, 8.9 } };
        //声明一个二维数组 values，并对其初始化
        foreach (double elementValue in values)
        Console.Write("{0} ", elementValue); //输出数组元素值
     }
}
```

foreach 语句只可循环访问数组或对象集合以读取所需信息，不能更改其中的内容，以避免产生不可预知的副作用。

3．跳转语句

用于转移控制。这一组语句有 break、continue、goto 和 return 语句。

（1）break 语句

格式：

```
break;
```

用于终止 switch 语句或中断当前正在执行的循环过程，转而执行循环结构外面的语句。

【例 6-6】判断数 m 是否为质数。

```
using System;
class Program
{
    static void Main()
    {
        int i, m;
        string s;
        m = Convert.ToInt32(Console.ReadLine());//键盘输入待判断的数
        for (i = 2; i <= Math.Sqrt(m); i++)
            if (m % i == 0) break;     //m 被 2～√m 中的数除尽，提前退出循环
        if (i > Math.Sqrt(m))
            s = m.ToString() + " is a prime number!"; //m 是质数
        else
            s = m.ToString() + " is not a prime number!"; //m 不是质数
        Console.Write(s);
     }
}
```

（2）continue 语句

格式：

```
continue;
```

立即终止本次循环的执行，忽视循环体中剩余语句，直接判断是否继续下一次循环。

（3）goto 语句

格式：

```
goto 标号;
```

将程序直接转向标号标记的语句，一般不推荐使用。

（4）return 语句

格式：

```
return [表达式];
```

终止包含它的函数执行，并将控制返回到主调函数继续执行。此处表达式为可选项，当不要求有返回值时，可以直接写 return。

4．异常处理

为了处理程序中可能产生的各种异常，C#为我们提供了 try…catch…finally 语句来捕获和处理异常。

格式：

```
try
{
    可能引发异常的程序代码块;
} catch (异常类型 1)
 {
    异常处理代码块 1;
 }
   ...
  catch (异常类型 n)
  {
    异常处理代码块 n;
  } finally
    {
       无论是否发生异常均要执行的代码块;
     }
```

try…catch…finally 语句捕获和处理异常的原理是：当 try 子句产生异常时，程序顺序查找第一个能处理该异常的 catch 子句，并使程序转向该 catch 子句。最后，不管是否发生异常，都执行 finally 子句的内容。

【例 6-7】文件读取的异常处理。

```
using System;
using System.IO;//使用文件必须引用该命名空间
public class Example
{   public static void Main()
{   StreamReader sr=null;//必须赋初值 null,否则编译不能通过
      try
    {  sr=File.OpenText("c:\\csharp\\test.txt");//打开文件可能产生异常
       string s;
       while(sr.Peek()!=-1)   // 文件非空时
         {   s=sr.ReadLine();   //读文件一行子串内容，可能产生异常
            Console.WriteLine(s);
          }
     }
    catch(DirectoryNotFoundException e) //无指定目录异常
        {    Console.WriteLine(e.Message);
        }
    catch(FileNotFoundException e)//无指定文件异常
       {    Console.WriteLine("文件"+e.FileName+"未被发现");
        }
    catch(Exception e) /其它所有异常
        {   Console.WriteLine("文件处理失败：{0}",e.Message);
        }
    finally
        {    if(sr!=null)   //若文件已打开则关闭文件
```

```
                sr.Close();
            }
        }
        }
```

通常情况是：在 try 块中获取并使用资源，在 catch 块中处理异常情况，并在 finally 块中释放资源。

6.2.5　C#常用系统类

在编写程序时，经常会用到一些操作，如字符串操作，数据类型转换等。C#语言将这些常见的功能以系统类的方式提供给用户，供编程时使用。下面列出的是一些常用的系统类。

1．数据转换

在 System 命名空间下，Convert 类提供了许多类型的静态转换方法，可以实现字符串与其他数据类型之间的转换。Convert 类的常用转换方法如表 6-9 所示。

表 6-9　System.Convert 类的常用转换方法

常 用 方 法	说　　明
ToInt32(数字字符串)	转换为 32 位有符号整数
ToInt64(数字字符串)	转换为 64 位有符号整数
ToSingle(数字字符串)	转换为单精度浮点数
ToDouble(数字字符串)	转换为双精度浮点数
ToDecimal(数字字符串)	转换高精度十进制数
ToDateTime(日期格式字符串)	转换为日期时间
ToChar(整型值)	转换为 Unicode 字符
ToString(各种类型数值)	转换为等效的字符串形式
ToBoolean(数字或字符串)	转换为等效的布尔值

例如：将字符串转换为整型数

```
string newString = "123456789";
int MyInt1 = Convert.ToInt32(newString);// MyInt1 的值为 123456789
Double MyDouble = 42.72;
int MyInt2 = Convert.ToInt32(MyDouble); // MyInt2 的值为 43
```

例如：将字符串转换为日期型或布尔型

```
string s1 = "2015-01-31 16:32:57" ;
DateTime b = Convert.ToDateTime(s1) ;//b 的值为 2015-01-31 16:32:57
string MyString = "true";
bool MyBool = Convert.ToBoolean(MyString); // MyBool 的值为 true
```

由于根类型 Object 具有 ToString()方法，因此各种类型也都继承了该方法，所以以下两种使用方法是完全等价的。

```
string d = Convert.ToString(i);
```

或

```
string d = i.ToString();
```

2．字符串操作

对字符串进行各种操作是程序中经常会使用的功能，字符串类有许多属性和方法可以实现相应的字符串处理。常用的属性和方法如表 6-10 所示。

表 6-10　System.String 类的常用属性和方法

常用属性和方法	说　明
Empty 静态属性	判定字符串是否为空串，取值为 true 和 false
Length 实例属性	返回字符串的长度
Compare(字串 1，字串 2)静态方法	比较两个字串的大小，取值为 1，0，-1。字串 1 大于字串 2，值为 1，相等时为 0，小于时为-1
Substring(start，length)	从字符串的 start 个字符位置开始截取 length 长度的字符子串
IndexOf(子串，起始位置)	从指定位置开始查找子串第一次出现的位置，第一个字符位置为 0。如果未找到搜索字符串，IndexOf()返回-1
Replace(字串 1，字串 2)	将字符串中的字串 1 替换成字串 2
Insert(插入位置，字串)	在指定位置插入一个字串
Trim()	删除字符串前后的空格
ToUpper()	将字符串中所有小写字符转换为大写字符
ToLower ()	将字符串中所有大写字符转换为小写字符

例如：

```
string s = "计算机科学与计算机技术";
int i1=s.Length;                          //串长 i1 为 11
string w1 = s.Substring(3,2);             //截取的子串为“科学”
int i2=s.IndexOf("机");                   //字符“机”第一次出现的位置为 2
string s1=s.Replace("计算机","软件");     //
//s1 内容为"软件科学与软件技术"，s 内容不变
int i3= string.Compare(s,s1);             //s 小于 s1，i3 的值为-1
bool flag=s==string.Empty;                // s 不是空串，flag 的值为 false
```

3．数组操作

在 System 命名空间下，Array 类提供了许多方法和属性，用于数组的创建、搜索、排序等操作。如表 6-11 所示。

表 6-11　System.Array 类的常用属性和方法

常用属性和方法	说　明
Length 属性	返回数组中所有元素的个数
Rank 属性	返回数组的维数
GetLength 属性	返回数组指定维中的元素个数
Sort()方法	用冒泡法对一维数组按照从小到大的顺序进行排序
Reverse()方法	将一维数组逆序
GetLowerBound()/GetUpperBound()方法	返回数组指定维的上限和下限
Copy()/CopyTo()方法	将数组的一部分复制到另一个数组中
Clear()方法	清除数组中所有元素的值，即将数组初始化。值类型设置为 0，引用类型设置为 null
GetValue()/SetValue()方法	获取/设置数组中指定元素值

【例 6-8】Array 类的使用。

```
using System;
class Program
{
    static void Main()
    {
```

```
            int[] m = new int[6] { 1, 6, 10, 0, 2, 55 };
              //定义一个一维数组 m 并初始化
            Console.WriteLine("输出数组 m 中的元素：");
            for (int i = 0; i < m.Length; i++)
                Console.WriteLine("m[{0}]={1}", i, m[i]);
            m.SetValue(99, 3);   //将下标为 3 的数组元素值置为 99
        Console.WriteLine("输出数组 m 值更改后的元素：");
            for (int i = 0; i <= m.GetUpperBound(0); i++)
                Console.WriteLine("m[{0}]={1}", i, m.GetValue(i));
            Array.Sort(m);//数组按从小到大顺序进行排序
    Console.WriteLine("输出数组 m 排序后的元素：");
            for (int i = 0; i <= m.GetUpperBound(0); i++)
                Console.WriteLine("m[{0}]={1}", i, m.GetValue(i));
        }
    }
```

程序运行结果如下：

输出数组 m 中的元素：

m[0]=1

m[1]=6

m[2]=10

m[3]=0

m[4]=2

m[5]=55

输出数组 m 值更改后的元素：

m[0]=1

m[1]=6

m[2]=10

m[3]=99

m[4]=2

m[5]=55

输出数组 m 排序后的元素：

m[0]=1

m[1]=2

m[2]=6

m[3]=10

m[4]=55

m[5]=99

4．数学运算

Math 类提供了许多算术运算的函数方法，如求平方根、求三角函数等。表 6-12 是常用的数学函数方法。

```
    double x1, x2;
    const double PI = 3.14159;
    x1 = Math.Abs(-45) ;                //变量 x1 的值为 45
    x2 = Math.Sin(60 * PI/ 180) ;       //变量 x2 的值为 0.866024961519134
```

注意：在使用 Math 类的方法时，须在方法名前加上类名 Math。

表 6-12　System.Math 类的常用方法

方　法　名	说　　明	举　　例	结　　果
Sqrt(x)	求 x 的平方根，x≥0	Math.Sqrt(16)	4
Abs(x)	求 x 的绝对值	Math.Abs(-5.3)	5.3
Exp(x)	求以 e 为底的指数幂，即 e^x	Math.Exp(2)	7.38905609893065
Log(x)	求以 e 为底的自然对数	Math.Log(15)	2.70805020110221
Sign(x)	符号函数，x 为正数，返回 1；x 为负数，返回 -1；x 为 0，返回 0	Math.Sign(2.3) Math.Sign(-2.3)	1 -1
Sin(x)	求 x 的正弦函数，x 单位是弧度	Math.Sin(3.141592/6)	0.499999905662436
Cos(x)	求 x 的余弦函数，x 单位是弧度	Math.Cos((3.141592/6)	0.86602545825025
Tan(x)	求 x 正切值，x 单位是弧度	Math.Tan(3.141592/6)	0.577350123947459
Atn(x)	求 x 反正切值	Math.Atan(3.141592/6)	0.482347821607832
Ceiling(x)	返回大于或等于指定数字的最小整数	Math.Ceiling(-2.1)	-2

5．日期和时间

对日期和时间的操作主要使用 System.DateTime 类，常用的属性和方法如表 6-13 所示。

表 6-13　System.DateTime 类的常用属性和方法

常用属性和方法	说　　明
Now 静态属性	取得系统当前的日期和时间
today 静态属性	取得系统当前的日期
Year()	取得给定 DateTime 值的年份
Mounth()	取得 DateTime 值的月份
Day()	取得 DateTime 值的日期
Hour()	取得 DateTime 值的小时数
Minute()	取得 DateTime 值的分钟数
Second()	取得 DateTime 值的秒数
DayOfWeek()	取得 DateTime 值的星期几
Add(时间段)	加上指定的时间段
AddDays(天数)	加上指定的天数

例如：

```
DateTime s = DateTime.Now;
// 取当前系统的日期和时间，2015-1-19 17:21:12
int i1 = s.Year; //当前年份 2015
int i2 = s. Hour; //当前时间 17 时
int i3 = s.AddDays(-7).Day; //7 天前的日期是 12
DayOfWeek s1 = s.DayOfWeek;//取今天的星期 Monday
```

6.2.6　C#面向对象的编程

面向对象编程（Object Oriented Programming，OOP，面向对象程序设计）是一种基于对象概念建立模型，模拟客观世界进行分析、设计、实现的编程架构，它解决了传统编程技术中存在的许多问题。C#是一种面向对象的编程语言，它的程序就是由用户自定义的类和.NETFramework 提供的类构成的。

1．类和对象

现实生活中的对象无处不在，一台电脑，一张桌子都是一个对象，它们分属于电子产品类和

家具类。一个类中可以包含（生成）多个对象，一个对象必定属于某一个类。所以，类是对同一种对象的统称，对象则是类的具体实例。在 C#中，类实际上是一种数据类型，可以用它来创建对象。

（1）类的声明

类的声明其实就是一种类型声明，用来定义一个新的数据类型。定义格式如下：

```
[访问修饰符] class <类名>
{
    类成员;
}
```

其中访问修饰符是任选项，可以是 public（公共的）、private（私有的）、protected（受保护的）和 internal（内部可访问的），默认为 internal。

【例 6-9】定义一个 Circle 类。

```
class Circle
{   const double PI=3.14; //类成员，一个常量
    private double radius;// 类成员，半径
    double area; //类成员，面积
    public void SetRadius(double _radius)// 类成员，设置半径
    {
        radius = _radius;
    }
    public void CalcArea()   //类成员，计算面积
    {
        area = PI * radius * radius;
    }
    public void Show()//类成员，输出数据
    {
        Console.WriteLine("radius="+radius);
        Console.WriteLine("area="+area);
    }
}
```

该类中声明了 6 个类成员，PI（常量）、radius（半径）和 area（面积）表示类的数据，SetRadius()、CalcArea()和 Show()表示对类数据的操作。

（2）创建对象和访问对象

声明了类之后，就可以通过 new 运算符来创建对象。创建类对象的格式如下：

```
类名 对象名=new 类名（参数）；
```

此处参数可以是 0 个或多个，多个参数之间用“，”隔开。

访问对象使用“.”操作符。

【例 6-10】创建 Circle 类（Circle 类定义见例 6-9）对象并访问 Circle 类对象。

```
class Program
{
        static void Main()
        {
            Circle circle=new Circle();//创建 Circle 类对象 circle
            circle.SetRadius(2);
            //调用 circle 对象的函数成员 SetRadius，设置圆的半径
            circle.CalcArea();
            //调用 circle 对象的函数成员 CalcArea，计算圆的面积
```

```
            circle.Show();
            //调用 circle 对象的函数成员 Show，输出圆的半径和面积
        }
    }
```

将例 6-9 和例 6-10 程序代码合并，运行，输出结果如下：

```
Radius=2
area=12.56
```

程序中的对象创建语句 Circle circle=new Circle()，还可以写成如下形式：

```
Circle circle;
circle=new Circle();
```

2．类的成员

类成员包括数据成员和函数成员。数据成员可以是字段、常量和事件。函数成员包括方法、属性、构造函数和析构函数等。所有的类成员通过访问修饰符限定访问权限。共有 5 种形式，含义如下所示。

① Public：公有的，在类外可以直接访问。

② Private：私有的，只能在本类内访问。

③ Protected：保护的，只能在本类内或派生类内访问。

④ Internal：可以被同一项目的代码访问。

⑤ protected internal：可以被同一项目的代码或继承类访问。

当缺省访问修饰符时，默认为 private。

（1）字段

与类或对象相关联的变量。定义方法与变量相同，并且可以进行初始化。用关键字 readonly 修饰时，表示该字段只能在构造函数中赋值，或由初始化语句赋值。字段的定义格式如下：

```
[访问修饰符] 数据类型 字段名；
```

例 6-9 中，类 Circle 声明了两个私有的字段 radius（半径）和 area（面积），其中字段 area 缺省访问修饰符，默认为私有的。

（2）方法

方法用于处理类的数据或执行某项操作。在 C#中，方法就是对象中的函数。方法的定义格式如下：

```
[访问修饰符] 返回类型 <方法名>（[参数]）
{
  语句组；
  [return 语句]
 }
```

当返回类型为 void 时，表示无返回值，此时可以省略 return 语句。方法名后的参数任选，可有 0 个或多个，多个参数之间用“，”隔开。无论有无参数小括号不可省略。

例 6-9 中，类 Circle 声明了三个方法，均是公有的，无返回值。其中 SetRadius()方法有一个参数，CalcArea()方法和 Show()方法不带参数。

方法中的参数有以下 4 种不同形式。我们把在方法定义中使用的参数称作形参，方法调用中使用的参数称做实参。

① 值参数

在参数声明时未加任何修饰符的形参就是值参数。调用方法时将实参值的副本存放在为值参数开辟的一个临时存储空间中，因而在方法中改变值参数的值不会影响到实参。值参数对应的实参可以是变量、常量和表达式。如例 6-9 中，SetRadius()方法有一个 double 类型的

值参数。

② 引用参数

若形参前加上修饰符 ref 就是引用参数。调用时将实参在内存的地址传递给形参，也就是实参、形参共用一个内存空间。所以在方法中改变形参的值就是改变对应实参的值。引用参数对应的实参必须是变量，不能是常量和表达式，在方法调用之前已经初始化。

③ 输出参数

用 out 修饰符声明的参数称为输出参数。输出参数与引用参数类似，与实参共用一个存储单元。这样，当在方法中给输出参数赋值时，就相当于给对应的实参赋值。它与引用参数的区别在于：引用参数可以实现双向数据传递，而输出参数只能向方法外传递数据，是单向的。传递给该方法的实参变量可以不初始化。

【例 6-11】比较值参数、引用参数、输出参数。

```
using System;
class Program
{
    public static void process(int x,ref int y,out int z)
    {
        x++; y++; z=3;
    }
    static void Main()
        {
            int a, b,c;
            a = 1;
            b = 1;
            process(a,ref b,out c);//注意方法 process 的调用格式
            Console.WriteLine("a="+a);
            Console.WriteLine("b="+ b);
            Console.WriteLine("c="+c);
        }
}
```

程序运行结果如下：

```
a=1
b=2
c=3
```

process()方法中有三个参数 x，y，z。其中参数 x 是值参数，改变参数 x 的值，不会影响对应实参 a 的值，所以 a 的值不变。参数 y 是引用参数，y 值的变化直接影响到实参 b，y 的值加 1，b 的值也加 1，变为 2。参数 z 是输出参数，尽管对应的实参 c 没有赋值，但在 process()方法中，对应的虚参 z 得到了值 3，所以 3 传给了 c，实参 c 的值也为 3。

④ 参数数组

参数数组允许向方法传递个数变化的参数，用修饰符 params 声明。参数数组只能是一维数组，如果形参表中包含参数数组，则该参数数组必须位于列表的最后。

【例 6-12】使用参数数组。

```
using System;
class Program
{
    public static void Sum(ref int sum, params int[] b)
//参数数组 b 放在列表的最后
    {
```

```
            for (int i=0;i<b.Length ;i++)
                sum=sum+b[i];
        }
        static void Main()
        {
            int []a=new int[]{1,2,3,4,5,6,7};
            int sum = 0;
            Console.WriteLine("输出数组元素的值");
            for (int i=0;i<a.Length ;i++)
                Console.WriteLine("a[{0}]={1}",i, a[i]);
            Sum(ref sum,a);//调用 Sum 方法求数组元素的和
            Console.WriteLine("数组元素的和为："+sum);
        }
    }
```

运行程序，结果如下：

输出数组元素的值

a[0]=1

a[1]=2

a[2]=3

a[3]=4

a[4]=5

a[5]=6

a[6]=7

数组元素的和为：28

（3）构造函数和析构函数

构造函数是一种特殊的类成员函数。通过它，在创建对象时就可直接初始化对象，如同普通变量的初始化。在创建对象时，构造函数自动被调用。其定义格式如下：

```
类名(<形参表>)
{
    …
}
```

构造函数具有以下几个特点：

① 函数名与类名相同；

② 没有返回类型，也不需要返回值；

③ 构造函数可以有 0 个或多个参数；

④ 构造函数可以重载；

⑤ 一般情况下，构造函数的访问权限是公有的。

【例 6-13】定义构造函数。

```
using System;
class Circle
{
    private double radius;// 字段，半径
    double area; //字段，面积
    public Circle(double _radius)// 带参数的构造函数
    {
        radius = _radius;
    }
    public Circle()// 无参构造函数
```

```
        {
        }
        public void SetRadius(double _radius)// 方法，设置半径
        {
            radius = _radius;
        }
        public void CalcArea()   //方法，计算圆的面积
        {
            area = 3.14 * radius * radius;
        }
        public void Show()//方法，输出数据
        {
            Console.WriteLine("radius=" + radius);
            Console.WriteLine("area=" + area);
        }
    }
    class Program
    {
        static void Main()
        {
            Circle circle1=new Circle();//调用无参构造函数创建对象，不初始化
            Circle circle2=new Circle(1);//调用有参构造函数创建和初始化对象
            circle1.SetRadius(2); //调用 SetRadius 方法，设置圆的半径
            circle1.CalcArea();
            circle1.Show();
            circle2.CalcArea();
            circle2.Show();
        }
    }
```

运行程序，结果如下：

Radius=2

area=12.56

Radius=1

area=3.14

上述程序中 Circle 类提供了两个重载的构造函数：一个是无参的构造函数，另一个是有参的构造函数。如果类中没有定义构造函数，如例 6-9，则编译器会自动产生一个缺省的、隐藏的、无参的默认构造函数，形式如同例 6-13 中的无参构造函数。如果自定义了构造函数，系统将不会产生缺省的默认构造函数，所以例 6-13 中，必须有一个自定义的无参构造函数。

与构造函数相对的是析构函数。创建对象时自动调用构造函数，撤销对象时自动调用析构函数。析构函数的定义格式如下：

```
~类名()
{    ...    }
```

析构函数特点如下：

① 与类名同名，函数名前加“~”；

② 不带参数，没有返回类型；

③ 一个类只有一个析构函数，不允许重载；

④ 如果类中没有定义析构函数，系统会自动生成一个。一般情况下，无须定义析构函数，除非撤销对象时涉及到释放内存或其他一些重要操作。

（4）属性

属性为访问类中的私有数据成员提供了安全保证，按照与方法类似的方式执行。属性的定义通过 get 和 set 关键字实现。set 是设置属性值的访问器，get 是获取属性值的访问器。如果只有 set 访问器，表示是只写属性；如果只有 get 访问器，表示是只读属性；如果同时具有这两个访问器，表示是读写属性。

【例 6-14】定义属性。

```
using System;
class Circle
{
        private double radius;// 字段，半径
        double area; //字段，面积
        public double Radius   //指定属性的类型和名称
        {
            get { return radius; }
            set { radius=value; }
        }
        public double Area //指定属性的类型和名称
        {
            get
            {
                area = 3.14 * radius * radius;
                return area;
            }
        }
}
class Program
{
        static void Main()
        {
            Circle circle = new Circle();
            circle.Radius= 2;//调用 set 访问器设置属性 Radius 值
            Console.WriteLine("radius=" + circle.Radius);
// 调用 get 访问器访问读写属性 Radius
            Console.WriteLine("area=" + circle.Area);
//调用 get 访问器访问只读属性 Area
         }
}
```

程序输出结果为：

radius=2

area=12.56

在 Circle 类中，声明了两个字段 radius 和 area。为了在类外访问这两个私有数据成员，相应的定义了两个公有属性，一个 double 型的读写属性 Radius，一个 double 型的只读属性 Area。属性名可以使用任何名称，为了表示更直观，一般将字段名的首字母大写作为属性名。关键字 value 代表 set 操作时用户提供的属性值。

（5）事件

事件就是对象或类状态改变时发出的消息或通知。引发事件的对象称为“发行者”，俘获事件并对其做出相应的对象称为“订户”。在典型的 C# Windows 窗体程序或 Web 应用程序中，许多事件是由控件（如按钮和列表框）引发的。C#中事件机制的工作过程如下。

① 将实际应用中需通过事件机制解决的问题对象注册到相应的事件处理程序上，表示今后当该对象的状态发生变化时，该对象有权使用它注册的事件处理程序。

② 当事件发生时，触发事件的对象就会调用该对象所有已注册的事件处理程序。

（6）静态成员与实例成员

类中成员有静态与实例之分。静态成员用 static 修饰符声明，它属于某个类而不属于某个具体的对象，使用时不需要建立类的对象，直接通过类名调用。无论类创建了多少实例，类的静态成员在内存中只占同一块区域。最熟悉的一个静态成员（静态方法）就是 Main()方法和 Console 类中的 WriteLine()方法。实例成员不用 static 修饰符声明，属于某个对象，使用时通过对象名调用。每创建一个类的实例(对象)，都会在内存中为非静态成员新分配一块区域存储。

【例 6-15】实例成员与静态成员的用法示例。

```
using System;
class Counter
{
    static int count;          //定义一个静态数据成员，初始值为 0
    int objNo;                 //定义一个非静态数据成员，表示对象编号
    public Counter()
    {
        count++;
        objNo=count;
    }
    public void Show1()
    {
        Console.WriteLine("obj=" + objNo);
        Console.WriteLine("count="+count);
    }
    public static void Show2()//定义一个静态函数成员
    {
        Console.WriteLine("count="+count);
    }
}
class Program
{
    static void Main()
    {
        Counter obj1=new Counter();
        obj1.Show1(); //调用非静态函数成员
        Console.WriteLine("----------");
        Counter obj2=new Counter();
        obj1.Show1();//调用非静态函数成员
        obj2.Show1();//调用非静态函数成员
        Counter.Show2(); //通过类名调用静态函数成员
    }
}
```

程序输出结果为：

```
Obj=1
count=1
----------
Obj=1
```

count=2
Obj=2
count=2
count=2

从运行结果中可以看出，对象 obj1 和对象 obj2 的 count 值是相等的，均为 2。这是因为 count 是静态成员，属于 Counter 类。在创建对象 obj1 时，将 count 值置 1。创建对象 obj2 时，使用的是同一个变量 count，值加 1 后，变成 2。这样不论是通过对象 obj1 还是对象 obj2 来引用值都是一样的。程序中，Show2()方法被定义为静态方法，所以直接通过类名调用。有一点要强调的是静态方法只能访问类中的静态成员，所以 Show2()方法只可输出静态变量 count 的值，而不可输出非静态变量 objNo 的值。

3．继承和多态

（1）继承

继承是面向对象程序设计的基本特征，它允许在已存在类的基础上建立新类。已存在的类称为基类，又称为父类，由已经存在的类派生出的新类称为派生类，又称为子类。例如，从类 A 中派生出类 B，则类 B 为类 A 的派生类，类 A 为类 B 的基类。派生类的定义格式如下：

```
[访问修饰符] class <派生类名>:<基类名>
{
    派生类新增加的成员;
}
```

C#继承有如下几个特点：

① C#只允许单继承，即生成的派生类只有一个基类；

② 继承具有传递性。如果类 A 派生出类 B，类 B 又派生出类 C，那么类 C 就拥有了类 A 和类 B 的所有成员；

③ 构造函数和析构函数不允许继承；

④ 基类的成员在派生类中不允许删除，但可以在派生类中声明同名的新成员，隐藏基类成员。

【例 6-16】从 Circle 类派生一个新类 Cylinder，求圆柱体的表面积和体积。

```
using System;
class Circle   //基类，圆
{   protected const double PI=3.14;//常量
    protected double radius;//  半径
    protected double area; //面积
    public void SetRadius(double _radius)//设置半径
    {
        radius = _radius;
    }
    public void CalcArea() //计算圆的面积
    {
        area = PI * radius * radius;
    }
    public void Show() //输出半径和面积
    {
        Console.WriteLine("radius=" + radius);
        Console.WriteLine("area=" + area);
    }
}
class Cylinder : Circle   //派生类，圆柱体
```

```
{
    private double high;//高度
    private double volume;//体积
    public void SetData(double _radius, double _high)//  设置圆柱体的高、半径
    {
        high =_high;
        radius=_radius;
    }
    public new void    CalcArea()//使用关键字重载 CalcArea
  //声明了一个和基类成员同名的新成员，计算圆柱体的表面积
    {
        area=2*PI*radius*radius+2*PI*radius*high;
    }
    public void CalcVolume() //计算圆柱体的体积
    {
        volume = PI * radius * radius * high;
    }
    public void    ShowVolume()//输出圆柱体的体积
    {
        Console.WriteLine("volume=" + volume);
    }
    public void ShowHigh()//输出圆柱体的高
    {
        Console.WriteLine("high=" + high);
    }
}
class Program
{
        static void Main()
        {
            Cylinder cylinder= new Cylinder();
            cylinder.SetData(2,5);
            cylinder.CalcArea();//计算圆柱体表面积
            cylinder.CalcVolume();//计算圆柱体体积
            cylinder.ShowHigh();//调用派生类的方法，输出圆柱体高
            cylinder.Show();//调用基类的方法，输出半径和面积
            cylinder.ShowVolume();//调用派生类的方法，输出体积
         }
}
```

程序运行结果如下：

high=5

radius=2

area=87.92

volume=62.8

派生类 Cylinder 继承了基类 Circle 的成员 PI（常量）、radius（半径）、area（面积）、SetRadius()、Show()，添加了新的成员 high（高）、volume(体积)、CalcVolume()、ShowHigh()、ShowVolume()，重载了 CalcArea()成员，用来计算圆柱体的表面积。需要注意的是：派生类中重载基类 CalcArea()成员时，应使用关键字 new。

（2）多态

多态指某类对象在接受同样的消息时，做出的响应不同，从而实现“一种接口，多种方法”

的技术。这里，“ 接受同样的消息 ”指调用名称相同的函数成员，“ 做出的响应不同 ”指函数实现的功能不同。C#支持两种多态性，一种是编译时多态性，编译器在对源程序进行编译时就可以确定所调用的是哪一个函数，编译时多态性通过重载来实现，如方法重载；另一种是运行时多态性，在程序运行过程中根据具体情况来确定调用哪一个函数，如虚方法。在类的方法前加上关键字 virtual，该方法就成为虚方法。通过在派生类中对虚方法重载，实现多态性。

【例 6-17】使用虚方法实现多态性。

```
using System;
class Animal //基类：动物类
{
        public virtual void Speak()   //虚方法
            {   Console.WriteLine("How does a Animal speak?");}
}
class Cat: Animal        //派生类：猫类
{
     public override void Speak()   //重载虚方法
      { Console.WriteLine("cat: miao!miao!");}
}
class Dog: Animal       //派生类：狗类
{
     public override void Speak()   //重载虚方法
      { Console.WriteLine("dog: wang!wang!");}
}
class Program
{
          static void Main()
          {
               Animal aninal=new Animal();
               Dog dog=new Dog();
               Cat cat=new Cat();
               aninal.Speak();
               aninal = dog;
               aninal.Speak();
               aninal = cat;
               aninal.Speak();
            }
}
```

程序运行结果：

How does a Animal speak?

dog: wang!wang!

cat: miao!miao!

虚方法重载要求方法名称、参数类型、参数个数，参数顺序、返回值类型都必须与基类的虚方法完全一致。在重载虚方法时，还必须加上 override 修饰符。

6.3 服务器控件

ASP.NET 为用户提供了一组使用方便、功能强大的服务器控件。它是一组可重用的组件或对象，是 ASP.NET 页面上能够被服务器代码访问和操作的控件。每个服务器控件都有自己的属性和方法，可以响应事件，是 Web 应用程序的重要元素。所有的服务器控件都有一个 Id 属性，它是服

务器端代码访问和操作该控件的唯一标识。除此之外，服务器控件还具有一个共同的属性 Runat="server"，这个属性标志着控件是在服务器端进行处理的。

6.3.1　服务器控件的分类

ASP.NET 服务器控件主要分为以下三种类型：HTML 服务器控件、Web 服务器控件和用户自定义服务器控件。其中 Web 服务器控件又分为标准服务器控件、验证控件、导航控件、数据控件、登录控件等。

1．HTML 服务器控件

HTML 服务器控件是以 HTML 标记为基础衍生出来的控件。它与 HTML 标记相比增加了两种属性：Id 和 Runat。在程序执行过程中可以动态地读取和修改其各种属性值，HTML 服务器控件最主要地是改变了页面设计的方法和数据提交的方式，为实现页面元素和编程逻辑的分离提供了便利。HTML 服务器控件类是在命名空间 Syetem.Web.UI.HtmlControls 中定义的。HTML 服务器控件的语法格式如下：

```
< 控件标记 Id ="控件名称" 属性 1=属性值 1 ... Runat="Server"/>
```

例如，输入密码的文本框控件：

```
<input Id ="Password1" type="password" Runat="Server"/>
```

2．Web 服务器控件

Web 服务器控件是针对 HTML 控件的不足而新增的控件, 相比 HTML 服务器控件具有更多的内置功能，增加了方法和事件驱动能力。它定义在 Syetem.Web.UI.WebControls 命名空间中。除了包括一些常见的按钮和文本框控件外，还增加了一些特殊用途的控件，如数据访问控件、日历控件等。Web 服务器控件的语法格式如下：

```
<asp:控件标记 Id ="控件名称" 属性 1="属性值 1" ...Runat="Server" />
```

或

```
<asp:控件标记 Id ="控件名称" 属性 1="属性值 1" …Runat="Server">
</ asp:控件标记>
```

例如，输入密码的文本框控件：

```
<asp:TextBox Id="TextBox1" TextMode="Password" Runat="Server"/>
```

或

```
<asp:TextBox Id="TextBox1" TextMode="Password" Runat="Server"></asp:TextBox>
```

3．用户自定义服务器控件

自定义控件被定义在命名空间 System.Web.UI.Control 或 Syetem.Web.UI.WebControls 中，是编程人员自行设计和开发的控件。它存放在扩展名为.ascx 的文件中，使用时只需要将它们集成进 ASP.NET 应用程序中。通过这个方法，用户不仅可以使用自己定义的控件，还可以很方便地使用第三方提供的现成控件，如图表工具和树形图等，且大部分控件都可以在网上免费下载，这为广大程序开发者高效、快速地开发 Web 程序提供了方便。

6.3.2　Web 服务器控件的属性、事件和方法

Web 服务器控件是 ASP.NET 的特定对象，采用事件驱动的编程模型，客户端触发的事件在服务器端处理。每一个控件都有它自己的属性、方法和事件。但不同的控件也可以有相同的属性、方法和事件。

1．Web 服务器控件的共有属性

控件属性是控件特性的描述，包括控件的外观特性和非可视化特性。共有属性指大多数控件所具有的属性，如标识控件的 ID 属性，表示大小的 Width 和 Height 属性等。Web 服务器控件的共有属性如表 6-14 所示。

表 6-14　Web 服务器控件共有属性

属　性	说　明	属　性	说　明
AccessKey	定义控件的加速键	Font-Names	控件使用字体的列表
BackColor	控件的背景颜色	Font-Size	字体的大小
BorderColor	控件的边框颜色	Font-Underline	字体是否使用下画线
BorderStyle	控件的边框样式	ForeColor	控件上文本的颜色
BoderWidth	控件的边框宽度	Height	控件的高度
CSSClass	控件使用的样式表类	TabIndex	控件的 Tab 键顺序
Enabled	指定控件能否被访问	Text	控件上显示的文本
Font-Bold	字体是否为粗体	ToolTip	设置控件的提示信息
Font-Name	控件上文本的字体	Visible	设置控件是否可见
Runat	属性值固定为 Server	Width	控件的宽度

Web 服务器控件的属性既可以在设计阶段通过属性窗口设置，也可以在运行阶段通过程序代码设置。下面的程序代码分别用两种方法设置 Label 控件的 Text 属性。

```
<Script Language="C#" Runat="Server" >
protected void Page_Load(object sender, EventArgs e)
  {
      Label2.Text = "这也是标签"; //在程序中设置 Label2 控件的 Text 属性
  }
</Script>
<!--在设计阶段设置 Label1 控件的 Text 属性-->
< asp:Label Id ="Label1" Text="这是标签" Runat="Server"/><br>
<!--在设计阶段没有设置 Label2 控件的 Text 属性-->
< asp:Label Id ="Label2" Runat="Server"/>
```

2．Web 服务器控件的方法

Web 服务器控件的方法主要实现一些特定的功能，如使控件获取焦点等，实质上就是函数和过程。服务器控件的常用方法如表 6-15 所示。

表 6-15　Web 服务器控件常用方法

方　法	说　明
ApplyStyleSheetSkin()	将页面样式表中定义的属性应用于该控件
DataBind()	将控件与某个数据源进行绑定
Dispose()	从内存中释放控件之前，给控件一个执行清除任务的机会
Focus()	把输入焦点设置为该控件
GetType()	获取当前实例的类型

如例 6-1 中 Button2_Click()事件过程中的语句：

```
usename.Focus();
```

3．Web 服务器控件事件和事件过程

Web 服务器控件事件是使某个控件进入活动状态的一种操作或动作。例如，按下某个键、单

击一下鼠标都可触发一个控件事件。在例 6-1 中，单击“登录”和“取消”按钮都会触发 Button 控件的鼠标单击事件。事件发生以后，如果有相应的事件处理过程，就会完成过程所要求的功能。如“取消”按钮的单击事件过程，将“用户名”输入框和“密码”输入框的内容清空，并将光标定位到“用户名”输入框上。

6.3.3　标准服务器控件

1．Label 控件（标签框）

Label 控件主要用于文本显示。

（1）语法格式。Label 控件的语法格式如下：

```
<asp:Label Id ="控件名称" Text="所要显示的文字" Runat="Server" />
```

或

```
<asp:Label Id ="控件名称" Runat="Server" >所要显示的文字</asp:Label>
```

（2）属性。Text 属性是 Label 控件最重要的属性，表示在控件上显示的文本，可以通过程序修改 Text 属性值。

【例 6-18】定义一个 Id 属性为 Label1、文字内容为标签控件、字体属性为楷体大字体、宽度属性为 160 像素，背景色为绿色的服务器控件。程序代码如下：

```
<asp:Label Id="Label1"　Text="标签控件"　Runat="Server"　BackColor="Lime"
    Font-Names="楷体_GB2312"　Font-Size="Larger"　Width="160px"/>
```

2．TextBox 控件（文本框）

TextBox 控件通常用来接收用户的输入信息，如文本、数字和日期等。默认情况下，TextBox 控件是一个单行文本框，只能输入一行内容。但可以通过修改控件 TextMode 属性，将文本框设置为多行或密码形式。

（1）语法格式。TextBox 控件的语法格式如下：

```
<asp:TextBox Id="控件名称"　Runat="Server"　AutoPostBack="True | False"
Columns="字符数目"　MaxLength="字符数目"　Rows="列数"
Text="字符串"　TextMode="SingleLine | Multiline | Password"　Wrap="True | False"
OnTextChanged="事件过程名"/>
```

（2）属性。除了前面介绍的共有属性外，TextBox 控件还有如表 6-16 所示的一些属性。

（3）事件。TextChanged()事件。当文本框中的内容改变并且按下回车键，或焦点改变到另一个控件时将会触发 TextChanged()事件。

表 6-16　TextBox 控件的属性说明

属　性	说　明
AutoPostBack	该属性为一个布尔值。当取值为 True 时，向服务器发送文本框的内容，如果和上次发送的内容不同，就会触发 TextChanged()事件。值为 False 时，不触发
Columns	设置文本框的显示宽度（单位：字符）
MaxLength	设置文本框中允许输入的最大字符数。当 TextMode 属性设置为 MultiLine 时，该属性无效
Rows	设置多行文本框的显示行数。本属性在 TextMode 属性设置为 MultiLine 时有效
Text	用于获取或设置文本框中的内容
TextMode	设置文本框的显示模式。共有三种取值： 1. SingleLine—只可以输入一行。默认为 SingleLine 2. PassWord—密码输入，输入的字符以*代替 3. MultiLine—可输入多行
Wrap	设定是否自动断行。本属性在 TextMode 属性设置为 MultiLine 时有效

3．Button 控件（命令按钮）

Button 控件是接收用户输入命令的提交按钮。单击该按钮会触发按钮的 Click()事件，并执行相应的事件过程。

（1）语法格式。Button 控件的语法格式如下：

```
<asp:Button Id="控件名称" Runat="Server" Text="按钮上的文字"
OnClick="事件过程名"  OnMouseOver="事件过程名"  OnMouseOut="事件过程名" />
```

（2）属性。Text 属性，按钮上显示的文字，用以提示用户进行何种选择。

（3）事件。

① Click()事件。用鼠标单击 Button 控件时触发。Click()事件的使用方法见例 6-1 中的 Button1_Click()和 Button2_Click()的事件过程。

② MouseOver()事件。当用户的光标进入按钮范围时触发。可以利用此事件完成诸如当光标移入按钮范围时，使按钮发生某种显示上的改变，用以提示用户可以进行选择了。

③ MouseOut()事件。当用户光标脱离按钮范围时触发。

4．RadioButton 控件（单选钮）

表示一个单选钮，用于从一组互斥的单选钮选项中选择一个。

（1）语法格式。RadioButton 控件的语法格式如下：

```
<asp:RadioButton Id="控件名称" Runat="Server" AutoPostBack="True | False"
Checked="True | False" GroupName="单选按钮组名称" Text="标识控件的文字"
TextAlign="Right|left" OnCheckedChanged="事件过程名"/>
```

（2）属性。RadioButton 控件常用的属性如表 6-17 所示。

表 6-17　RadioButton 控件常用属性

属　性	说　明
AutoPostBack	当按钮状态改变时决定页面是否被传回。属性值为 True 时，传回；值为 False 时，不传回
Checked	设置或获取按钮的当前状态。选中时，Checked 值为 True。未选中时，值 False
GroupName	设置单选按钮组的名称。同组中的按钮只能选中一个
TextAlign	设置文本的位置是在按钮的左边或右边，默认 Right
Text	单选按钮边所显示的文本

（3）事件。CheckedChanged()事件，当单选钮的状态发生变化时触发该事件，前提是 AutoPostBack 属性值为 True。否则，该事件将被延迟。

5．RadioButtonList 控件（单选钮列表）

RadioButtonList 控件由一组 RadioButton 控件组成。这些按钮自动包含在一个组中，并且一组只能选中一个选项。使用 RadioButtonList 控件比使用多个 RadioButton 控件要方便得多。若要将按钮绑定到数据源，只能使用 RadioButtonList 控件。

（1）语法格式。RadioButtonList 控件的语法格式如下：

```
<asp:RadioButtonList Id="控件名称" Runat="Server" AutoPostBack="True | False"
    CellPadding="像素" CellSpacing="像素"
    RepeatDirection="Vertical | Horizontal"
    RepeatLayout="Flow | Table" TextAlign="Right | Left"
    RepeatColumns="列表的列数"
    OnSelectedIndexChanged="事件过程名">
    <asp:ListItem value="选项值 1" selected=" True | False " text="选项文字 1"/>
    <asp:ListItem value="选项值 2" selected=" True | False " text="选项文字 2"/>
```

```
        …
    </asp:RadioButtonList>
```

（2）属性。RadioButtonList 控件常用的属性如表 6-18 所示。

表 6-18　RadioButtonList 控件常用属性

属　性	说　明
AutoPostBack	决定更改 RadioButtonList 控件中的内容时，是否自动回送到服务器。默认值为 False，不回送
CellPadding	表示单元格的边框和内容之间的距离，单位是像素点数（px）
CellSpacing	表示单元格和单元格之间的距离，单位是像素点数（px）
Items	表示 RadioButtonList 控件中各选项的集合。如 RadioButtonList1.Items(*i*)表示第 *i* 个选项，*i* 从 0 开始。每个选项都有三个基本属性。①Text 属性—表示每个选项的文本。②Value 属性—表示每个选项的选项值。③Select 属性—表示该选项是否选中
RepeatColumns	设置列表使用的列数
RepeatDirection	设置 RadioButtonList 控件的排列方式。当属性值为 Horizontal 时，各选项以行优先排列；当属性值为 Vertical 时，各选项以列优先排列
RepeatLayout	设置 RadioButtonList 控件的排列方式。属性值为 Table 时，以一个不可见的表结构形式显示；属性值为 Flow 时，不以表结构显示
SelectedIndex	获取控件中选定项的索引值。第一项值为 0
SelectedItem	获取控件中选定项的 Text 属性值
SelectedValue	获取控件中选定项的 Value 属性值
TextAlign	设置显示文本的位置是在按钮的左边或右边，默认 Right

【例 6-19】单选钮列表控件示例。如图 6-4 所示，4 个选项对应的单选钮列表控件的设置代码如下：

```
    <asp:RadioButtonList ID="RadioButtonList1" Runat="Server" Height="77px"
        Width="141px" CellPadding="1" CellSpacing="1" RepeatColumns="2">
        <asp:ListItem Selected="True" Value="1">篮球</asp:ListItem>
        <asp:ListItem Value="2">排球</asp:ListItem>
        <asp:ListItem Value="3">乒乓球</asp:ListItem>
        <asp:ListItem Value="4">羽毛球</asp:ListItem>
    </asp:RadioButtonList>
```

（3）事件。SelectedIndexChanged()事件。当用户选择了控件中的某一选项时触发该事件。

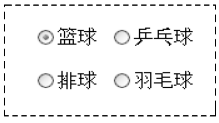


图 6-4　单选钮列表控件示例

6．CheckBox 控件（复选框）

CheckBox 控件用于在页面中创建复选框，允许用户从一组可选项中同时选中多个选项。

（1）语法格式。CheckBox 控件的语法格式如下：

```
    <asp:CheckBox Id="控件名称" Runat="Server" AutoPostBack="True | False"
    Text="标识控件的文字" TextAlign="Right|Left " Checked="True | False"
    OnCheckedChanged="事件过程名"/>
```

（2）属性。CheckBox 控件常用的属性如表 6-19 所示。

表 6-19　CheckBox 控件常用属性

属　性	说　明
AutoPostBack	决定单击 CheckBox 控件时是否自动回送到服务器
Checked	设置或获取复选框的选中状态。值为 True 时，表示选中。值为 False 时，表示未选中
Text	设置或获取复选框的标识文本
TextAlign	设置显示文本的位置是在复选框的左边或右边，默认 Right

（3）事件。CheckedChanged()事件。当复选框的状态发生变化时触发。如果 AutoPostBack 的值是 False，这个事件将被延迟触发。

7．CheckBoxList 控件（复选框列表）

CheckBoxList 控件由一组 CheckBox 控件组成，可以同时选中多个选项，主要用于数据绑定。

（1）语法格式。CheckBoxList 控件的语法格式如下：

```
<asp:CheckBoxList Id="控件名称" Runat="Server" AutoPostBack="True | False"
    CellPadding="像素" RepeatColumns="列表的列数"
    RepeatDirection="Vertical|Horizontal"
    RepeatLayout="Flow|Table"
    TextAlign="Right|Left"
    OnSelectedIndexChanged="事件过程名">
    <asp:ListItem value="选项值 1" selected=" True | False " text="选项文字 1" />
    <asp:ListItem value="选项值 2" selected=" True | False " text="选项文字 2" />
    …
</asp:CheckBoxList>
```

（2）属性。各属性的含义参见表 6-18 RadioButtonList 控件的常用属性。

（3）事件。SelectedIndexChanged()事件。当复选列表框中的选项改变时触发该事件。

8．DropDownList 控件（下拉列表框）

DropDownList 控件用于在页面中创建一个下拉列表，供用户从中选择一个选项。

（1）语法格式。DropDownList 控件的语法格式如下：

```
<asp:DropDownList Id="控件名称" Runat="Server" AutoPostBack="True | False"
    OnSelectedIndexChanged="事件过程名">
    <asp:ListItem value="选项值 1" selected=" True | False " text="选项文字 1"/>
    <asp:ListItem value="选项值 2" selected=" True | False " text="选项文字 2"/>
    …
</asp:DropDownList>
```

（2）属性。DropDownList 控件常用的属性如表 6-20 所示。

表 6-20　DropDownList 控件常用属性

属　性	说　明
AutoPostBack	当用户更改选项内容时，决定页面是否回传。值为 True，回传，触发 SelectedIndexChanged()事件；值为 False，不回传
Items	表示 DropDownList 控件中各选项的集合。如 ListBox1.Items(*i*)表示第 *i* 个选项，*i* 从 0 开始。每个选项都有三个基本属性。①Text 属性—表示该选项的文本。②Value 属性—表示该选项的选项值。③Select 属性—表示该选项是否选中。此外，Items 集合还有一个 Count 属性，表示 DropDownList 控件中选项的数目
SelectedIndex	获取所有选中选项的最小索引值。若未选定任何项，则返回值为-1
SelectedItem	获取列表中具有最小索引值的选定项。通过该属性可获得选定项的 Text 属性值

（3）事件。SelectedIndexChanged()事件。当用户在列表中进行选择时触发该事件。

（4）方法。

① Add()方法：将选项添加到 DropDownList 控件的列表末尾。

② RemoveAt()方法：删除 DropDownList 控件中指定的选项。

③ Clear()方法：清除 DropDownList 控件中的所有选项。

④ Insert()方法：将一个新的选项插入到 DropDownList 控件的指定位置。

【例 6-20】在一个下拉列表框控件中添加 4 个选项，并将第一项设置为默认选项。界面如图 6-5 所示。单击“插入一项”按钮，插入“西瓜”选项。单击“删除一项”按钮，删除列表中的第二项。单击“清除所有”按钮，清空列表项。程序代码如下：

图 6-5　下拉框控件示例

```
<%@ Page Language="C#" %>
<script runat="server">
protected void Page_Load(object sender, EventArgs e)
{
        if (!Page.IsPostBack)
        {
            DropDownList1.Items.Add("香蕉");
            DropDownList1.Items.Add("苹果");
            DropDownList1.Items.Add("梨子");
            DropDownList1.Items.Add("菠萝");
            DropDownList1.Items[0].Selected=true;
}
}
protected void Button1_Click(object sender, EventArgs e)
{
    DropDownList1.Items.Insert(0, "西瓜"); //将西瓜添加到列表中使其成为第一项
}
protected void   Button2_Click(object sender, EventArgs e)
{
    DropDownList1.Items.RemoveAt(1);      //删除列表中的第二项，香蕉被移除
}
protected void   Button3_Click(object sender, EventArgs e)
{
    DropDownList1.Items.Clear();          //清除所有项
}
</script>
<html>
<head runat="server">
    <title>无标题页</title>
</head>
<body>
    <form id="form1" runat="server">
      <div>
          <asp:DropDownList ID="DropDownList1" runat="server"/><p></p>
          <asp:Button ID="Button1" runat="server" Text="插入一项" onclick="Button1_Click" />
          <asp:Button ID="Button2" runat="server" Text="删除一项" onclick="Button2_Click" />
          <asp:Button ID="Button3" runat="server" Text="清除所有" onclick="Button3_Click" />
      </div>
    </form>
</body>
</html>
```

9．ListBox 控件（列表框）

ListBox 控件在页面中创建一个列表框，供用户从中选择一个或多个选项，与控件 DropDownList 具有许多相似的属性和方法。不同之处在于 ListBox 控件的列表框不隐藏，始终展开。

（1）语法格式。ListBox 控件的语法格式如下：

```
<asp:ListBox Id="控件名称" Runat="Server" AutoPostBack="True | False"
Rows="一次能显示的行数"   SelectionMode="Single | Multiple"
OnSelectedIndexChanged="事件过程名">
      <asp:ListItem value="选项值 1" selected="true|false" text="选项文字 1"/>
      <asp:ListItem value="选项值 2" selected="true|false" text="选项文字 2"/>
       …
</asp:ListBox>
```

（2）属性。ListBox 控件的许多属性与 DropDownList 控件相同，表 6-21 列出的是 ListBox 控件独有的属性。

（3）事件。与 DropDownList 控件一样也有一个 SelectedIndexChanged()事件。

（4）方法。同 DropDownList 控件。

表 6-21　ListBox 控件的属性

属　　性	说　　明
Rows	设定 ListBox 控件所能显示的列表项行数
SelectionMode	设定 ListBox 控件是否可以按住 Shift 或 Ctrl 键进行多选。默认值为 Single，单选

【例 6-21】一个模拟选课的案例，如图 6-6 所示。在页面中用到了一个普通文本框控件、一个密码文本框控件、一个 RadioButtonList 控件、一个 DropDownList 控件、一个 ListBox 控件、一个 CheckBoxList 控件、两个 Button 控件、一个多行文本框控件以及一个用于显示选课信息的 Label 控件。其中 DropDownList 控件（院）与 ListBox 控件（系）实现下拉联动。

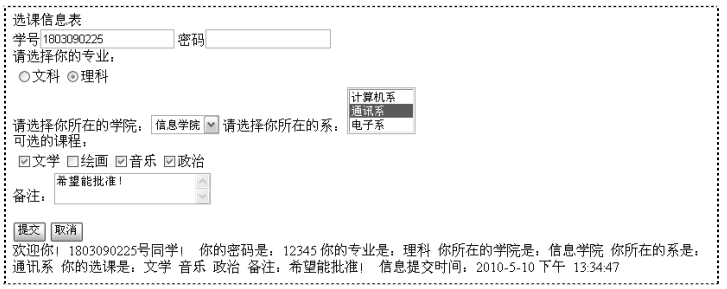


图 6-6　模拟选课案例

程序代码如下：

```
<%@ Page Language="C#" %>
<script runat="server">
void Button1_Click(object sender, EventArgs e)
{
      string s,temp ;
      int i;
      s = "欢迎你！" + Textnum.Text + "号同学！" + " ";
      s = s+ "你的密码是：" +Textpassword.Text + "    ";
      s = s + "你的专业是：" + Radiodepart.SelectedItem.Text + " ";
      s = s + "你所在的学院是：" + DropDowncollege.SelectedItem.Text + " ";
      s = s + "你所在的系是：" + Listdepart.SelectedItem.Text + " ";
      temp = "";
       //向 temp 加入选中的选课项目
      for(i = 0;i<=CheckBoxcourse.Items.Count-1;i++)
            if (CheckBoxcourse.Items[i].Selected)
                  temp = temp + CheckBoxcourse.Items[i].Text + " ";
      if (temp != "")    s = s + "你的选课是：" + temp +" ";
```

```
        if (Textmem.Text != "")    s = s + "备注：" + Textmem.Text + " ";
        labelmessage.Text = s +"信息提交时间："+ DateTime.Now +" ";
}
void    Button2_Click(object sender, EventArgs e)
{
                Textnum.Text = "";
                Textpassword.Text = "";
                labelmessage.Text = "";
}
void    selchange(object sender, EventArgs e)
{
          string snum ;
          snum = DropDowncollege.SelectedValue;
          Listdepart.Items.Clear();
          switch (snum)
          {
               case "0":
                    Listdepart.Items.Add("计算机系");
                    Listdepart.Items.Add("通讯系");
                    Listdepart.Items.Add("电子系");
                    break;
               case "1":
                    Listdepart.Items.Add("测绘测量系");
                    Listdepart.Items.Add("土木工程系");
                    Listdepart.Items.Add("地下工程系");
                    break;
               case "2":
                    Listdepart.Items.Add("建筑设计系");
                    Listdepart.Items.Add("建筑规划系");
                    break;
          }
          Listdepart.Items[0].Selected = true;
}
</script>
<html>
<head runat="server">
     <title>无标题页</title>
</head>
<body>
    <form id="form1" runat="server">
     <div>
          <asp:Label ID="Label1" runat="server" Text="选课信息表"></asp:Label>
          <br />
          学号<asp:TextBox ID="Textnum" runat="server"></asp:TextBox>
          密码<asp:TextBox ID="Textpassword" runat="server"
               TextMode="Password"></asp:TextBox>
          <br />
          请选择你的专业：<asp:RadioButtonList ID="Radiodepart" runat="server"
               RepeatDirection="Horizontal">
               <asp:ListItem Selected="True">文科</asp:ListItem>
               <asp:ListItem>理科</asp:ListItem>
          </asp:RadioButtonList>
```

```
            请选择你所在的学院：<asp:DropDownList ID="DropDowncollege" runat="server"
            autopostback="true" OnSelectedIndexChanged ="selchange">
                <asp:ListItem Value="0" selected="true">信息学院</asp:ListItem>
                <asp:ListItem Value="1">土木学院</asp:ListItem>
                <asp:ListItem Value="2">建筑学院</asp:ListItem>
             </asp:DropDownList>
            请选择你所在的系：<asp:ListBox ID="Listdepart" runat="server" Rows="3"></asp:ListBox>
            <br />
            可选的课程：<asp:CheckBoxList ID="CheckBoxcourse" runat="server"
                RepeatDirection="Horizontal">
                <asp:ListItem>文学</asp:ListItem>
                <asp:ListItem>绘画</asp:ListItem>
                <asp:ListItem>音乐</asp:ListItem>
                <asp:ListItem>政治</asp:ListItem>
            </asp:CheckBoxList>
            备注：<asp:TextBox ID="Textmem" runat="server" TextMode="MultiLine"/>
            <br />
            <asp:Button ID="Button1" runat="server" onclick="Button1_Click" Text="提交" />
            <asp:Button ID="Button2" runat="server" onclick="Button2_Click" Text="取消" />
            <br />
            <!--Label  显示用户信息的标签-->
            <asp:Label ID="labelmessage" runat="server" ></asp:Label>
          </div>
    </form>
</body>
</html>
```

10．Image 控件（图像）

Image 控件用于在页面上显示图片。

（1）语法格式。Image 控件的语法格式如下：

```
<asp:Image Id="控件名称" Runat="Server" ImageUrl="图片存放的路径"
AlternateText="提示文本"   ImageAlign="NotSet|AbsBottom|Absmiddle|BaseLine|Left|
Middle|Right|TextTop|Top"/>
```

（2）属性。Image 控件常用的属性如表 6-22 所示。

表 6-22　Image 控件常用属性

属　性	说　明
ImageUrl	设置显示图片的存储路径，可以是绝对路径，也可以是相对路径
AlternateText	将鼠标放在图像上所显示的提示文本。图像不可用时，作为替换文本
ImageAlign	设置图片相对于其他元素的对齐方式

11．ImageButton 控件（图像按钮）

ImageButton 控件与 Image 控件几乎一样，只是增加了一个鼠标单击事件。

（1）语法格式。ImageButton 控件的语法格式如下：

```
<asp:ImageButton Id="控件名称" Runat="Server" ImageUrl="图片的存放路径"
AlternateText="提示文本"   ImageAlign="NotSet|AbsBottom|Absmiddle|BaseLine|Left|
Middle|Right|TextTop|Top" onClick="事件过程名"/>
```

（2）属性。属性参见表 6-22 Image 控件常用属性。

（3）事件。Click()事件。当单击 ImageButton 控件按钮时触发。

12．HyperLink 控件（超链接）

使用 HyperLink 控件可以实现到其他网页的超链接，类似于 HTML 的<a herf="…">标记。在程序中，修改其属性可以动态修改链接文本和目标网址。

（1）语法格式。HyperLink 控件的语法格式如下：

```
<asp:HyperLink Id="控件名称" Runat="Server" NavigateUrl="链接的地址"
Text="超链接的文字"  Target="目标页的显示位置"
ImageUrl="图片存放的路径"/>
```

（2）属性。HyperLink 控件的常用属性如表 6-23 所示。

表 6-23　HyperLink 控件常用属性

属　性	说　明
NavigateUrl	指定单击控件时要链接的目标地址
Text	设置超链接的显示文字
Target	设置目的页面被打开的目标窗口和框架。常用属性值及含义如下： _Blank — 在一个新的窗口打开目的页面 _Self — 在当前窗口打开目的页面 _Parent — 在父窗口打开目的页面 _Top — 在最上层窗口打开目的页面
ImageUrl	指定一个图片文件，使 HyperLink 控件的外观显示为该图片。若同时还设置了 Text 属性，则优先显示图片

13．LinkButton 控件（超链接按钮）

LinkButton 控件用于创建类似于超级链接的按钮，它具有与 HyperLink 控件相同的外观，与 Button 控件完全相同的功能。它与 HyperLink 控件的区别在于：HyperLink 控件能自动导航到指定的目标页面，而 LinkButton 控件没有 NavigateUrl 属性，它会触发服务器端的 Click()事件。

（1）语法格式。LinkButton 控件的语法格式如下：

```
<asp:LinkButton Id="控件名称" Runat="Server" Text="按钮上的文字"
Onclick="事件过程名"/>
```

（2）属性。Text 属性用于设置或获取按钮上的显示文本。

（3）事件。Click()事件。与 Button 控件相同，当单击 LinkButton 控件时，触发该控件的 Click()事件。

14．Table 控件（表）

Table 控件创建一个具有行和列的表。Table（表）由 TableRow（表中的行）对象组成，而 TableRow 又由 TableCell（表中的单元格）对象组成。

（1）语法格式。Table 控件的语法格式如下：

```
<asp:Table Id="控件名称" Runat= "Srever" CellPadding="像素"
CellSpacing="像素" GridLines="None|Horizontal|Vertical|Both"
HorizontalAlign="Center|Justify|Left|NotSet|Right">
        <asp:TableRow>
        <asp:TableCell>...</asp:TableCell>
          ...
        </asp:TableRow>
          …
</asp:Table>
```

（2）属性。Table 控件的常用属性如表 6-24 所示。TableRow 对象常用属性和 TableCell 对象常用属性如表 6-25 和表 6-26 所示。

表 6-24　Table 控件常用属性

属　性	说　明
CellPadding	设置单元格的边框和内容之间的距离。单位：像素
CellSpacing	设置表中单元格之间的距离。单位：像素
Rows	获取表中行的集合。Rows 集合包含如下属性和方法： ① Count 属性—Rows 集合中元素的个数，即表的行数 ② Add()方法—添加一个新的 TableRow 对象，即在表格的末尾添加一个新行 ③ AddAt()方法—在指定位置插入一个 TableRow 对象，即插入一个新行到表的指定位置 ④ Remove()方法—删除一个 TableRow 对象，即从表中删除一行 ⑤ RemoveAt()方法—删除指定位置的 TableRow 对象，即删除指定位置的行 ⑥ Clear()方法—清除集合中所有的 TableRow 对象，即删除表中所有的行
GridLines	指定表中显示的网格线的样式。None—不显示单元格边框，Horizontal—只显示水平边框，Vertical—只显示垂直边框，Both—同时显示水平和垂直边框
HorizontalAlign	设置表格的水平对齐方式。Center—居中，Justify—左右边距对齐，Left—左对齐，NotSet—未设置，Right—右对齐

表 6-25　TableRow 对象常用属性

属　性	说　明
HorizontalAlign	设置行的水平对齐方式。可以是 Center、Justify、Left、NotSet、Right
VerticalAlign	设置行的垂直对齐特性。可以是 Middle、Top、Bottom
Cells	获取表中某行单元格的集合。Cells 集合的主要属性和方法如下： ① Count 属性—Cells 集合中元素的个数，即表的列数 ② Add()方法—添加一个新的 TableCell 对象，即在表中添加一个新的单元格 ③ AddAt()方法—在指定位置插入一个 TableCell 对象，即插入一个新的单元格到指定位置 ④ Remove()方法—删除一个 TableCell 对象，即从表中删除一个单元格 ⑤ RemoveAt()方法—删除指定位置的 TableCell 对象，即删除指定位置的单元格 ⑥ Clear()方法—清除集合中所有的 TableCell 对象，即删除表中某行的所有单元格

表 6-26　TableCell 对象常用属性

属　性	说　明
ColumnSpan	指定一个给定的单元格可以跨多少列
RowSpan	指定一个给定的单元格可以跨多少行
VerticalAlign HorizontalAlign	指定单元格的垂直和水平对齐属性
Wrap	指定单元格的内容是否允许换行。默认值为 False，不允许换行
Text	单元格中的文本

【例 6-22】Table 控件应用示例。本例中单击“ 隐藏表格 ”按钮（按钮 1），可以显示或隐藏一个 4×4 的表格，如图 6-7 所示。该表格由程序动态产生。单击“ 修改表格 ”按钮（按钮 2），将该表修改成一个 3×3 的表格，如图 6-8 所示。单击“ 清除表格 ”按钮（按钮 3），清除整个表格。程序代码如下：

图 6-7　产生一个 4×4 表格

图 6-8　修改成一个 3×3 表格

```
<%@ Page Language="C#" %>
<script runat="server">
  int    rowsnum ,cellsnum;
  int i,j,k;
void Button1_Click(object sender, EventArgs e)
{
        if (Button1.Text == "显示表格" )
        {
                Table1.Visible = true;
                Button1.Text = "隐藏表格";
                createtable();
        }
        else
        {
                Table1.Visible = false;
                Button1.Text = "显示表格";
        }
  }
void createtable()//创建表格的过程
{
            rowsnum = 4;
            cellsnum = 4;
            k = 0;
            for (i = 0 ;i<= rowsnum - 1;i++)            //创建一个 4 行 4 列的表格，内容为 1-16 的数字
            {
  TableRow r = new TableRow();                          //建立一个新行
                for (j = 0 ;j<=cellsnum - 1;j++)
                {
                    TableCell c = new TableCell();      //建立一个单元格
                    k = k + 1;
                    c.Text =k.ToString();               //向新单元格中填数据
                    r.Cells.Add(c);                     //将单元格添加到行中
                }
                Table1.Rows.Add(r);                     //将行添加到表中
            }
}
void    Button2_Click(object sender, EventArgs e)
{
      if (Button1.Text == "隐藏表格")
      {
            createtable();
            //修改表格使其成为 3 行 3 列
            Table1.Rows.RemoveAt(rowsnum - 1);          //移去最后一行
            for (i = 0; i <= Table1.Rows.Count - 1; i++)
            {
                TableCell c = Table1.Rows[i].Cells[cellsnum - 1];       //选取每一行的最后一列
                Table1.Rows[i].Cells.Remove(c);         //移去每一行的最后一列
            }
      }
}
void Button3_Click(object sender, EventArgs e)
{
```

```
            if ( Button1.Text == "隐藏表格")
                Table1.Rows.Clear();              //清除表中的所有行
    }
    </script>
    <html>
    <head runat="server">
        <title>无标题页</title>
    </head>
    <body>
        <form id="form1" runat="server">
          <div>
              <asp:Table ID="Table1" runat="server" CellPadding="1" CellSpacing="1"
                    GridLines="Both">
                </asp:Table>
              <asp:Button ID="Button1" runat="server" Text="显示表格" onclick="Button1_Click"/>
              <asp:Button ID="Button2" runat="server" Text="修改表格" onclick="Button2_Click"/>
              <asp:Button ID="Button3" runat="server" Text="清除表格" onclick="Button3_Click"/>
            </div>
        </form>
    </body>
    </html>
```

15．Panel 控件（面板）

Panel 控件是一个容器控件，可以将多个控件组合在一起。组合在一起的控件可以整体隐藏或显示，方法是将 Panel 控件的 Visible 属性设置为 False 或 True。

（1）语法格式。Panel 控件的语法格式如下：

```
<asp:Panel Id="控件名称"   Runat="Server" BackImageUrl="图片存放路径"
HorizontalALign="Center|Justify|Left|NotSet|Right" Visible="True|False"
Wrap="True|False" />
```

（2）属性。Panel 控件的常用属性如表 6-27 所示。

表 6-27　Panel 控件常用属性

属　性	说　明
BackImageUrl	背景图片的存放路径
HorizontalALign	设置面板中控件的水平对齐方式
Visible	设置 Panel 控件及其上控件是否显示在网页中。值为 True（默认值）时，显示；值为 False 时，不显示
Wrap	设置该控件中的内容是否允许换行。值为 True（默认值）时，自动换行；值为 False 时，不换行

16．Calendar 控件（日历）

Calendar 控件是一个日历控件，用于在浏览器中显示日历或选择日期。显示日历时，用户可以选择日期并可转到前、后月份。Calendar 控件具有很多属性，用来设置日历显示的外观、样式等特征。

（1）语法格式。Calendar 控件的语法格式如下：

```
<asp:Calendar Id="控件名称"   Runat="Server"
CellPadding="pixels" CellSpacing="pixels"
DayNameFormat="FirstLetter|FirstTwoLetters|Full|Short"
FirstDayOfWeek="Default|Monday|Tuesday|Wednesday| Thursday|Friday|Saturday|Sunday" >
</asp:Calendar>
```

（2）属性。Calendar 控件的常用属性如表 6-28 所示。

表 6-28　Calendar 控件常用属性

属　　性	说　　明
CellPadding	单元格边框与内容之间的距离，单位：像素
CellSpacing	单元格之间的距离，单位：像素
DayNameFormat	显示周中各天的名称格式
FirstDayOfWeek	指定每周的第一天

（3）事件。SelectionChanged()事件。当在 Calendar 控件上选择日期时触发。

【例 6-23】Calendar 控件使用。在标签控件中显示选中的日期。运行结果如图 6-9 所示。

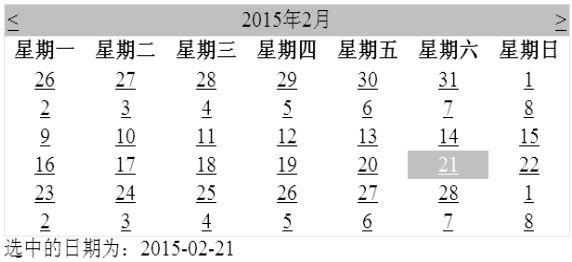


图 6-9　Calendar 控件使用

程序代码如下：

```
<%@ Page Language="C#" AutoEventWireup="true" %>
  <script runat="server">
  protected void Calendar1_SelectionChanged(object sender, EventArgs e)
{
      Label1.Text = "选中的日期为：" + Calendar1.SelectedDate.ToString("yyyy-MM-dd");
}
  </script>
<html>
<head runat="server">
      <title>无标题页</title>
</head>
<body>
      <form id="form1" runat="server">
      <div style="height: 190px">
            <asp:Calendar ID="Calendar1" runat="server" DayNameFormat="Full"
                  FirstDayOfWeek="Monday" onselectionchanged="Calendar1_SelectionChanged"
                  Width="610px"></asp:Calendar>
            <asp:Label ID="Label1" runat="server" Text="Label"></asp:Label>
            <br />
      </div>
      </form>
</body>
</html>
```

17．FileUpload 控件（文件上传）

使用 FileUpload 控件可以将客户端的文件上传到服务器端指定的目录下。该控件不仅可以上传图片、文本文件，还可以上传其他任意类型的文件。

（1）语法格式。FileUpload 控件的语法格式如下：

```
<asp:FileUpload ID="控件名称" runat="server" />
```

（2）属性。FileUpload 控件的常用属性如表 6-29 所示。

表 6-29　FileUpload 控件常用属性

属　性	说　明
FileName	上传文件的名称,不包含路径信息
FileContent	返回一个上传文件的流对象
HashFile	指示 FileUpload 控件中是否有要上传的文件，True 表示有，False 表示没有
PostedFile	获取上传文件的引用，属性值类型为“ HttpPostedFile ”

HttpPostedFile 对象的常用属性如表 6-30 所示。

表 6-30　HttpPostedFile 常用属性

属　性	说　明
ContentLength	上传文件的大小，用字节表示
ContentType	上传文件的 MIME 内容类型
FileName	上传文件的文件名，包含客户端路径信息
InputStream	返回上传文件的流对象

（3）方法。FileUpload 控件的常用方法是 SaveAs()，将上传的文件保存到服务器指定的文件目录中。

【例 6-24】利用 FileUpload 控件上传文件。运行界面如图 6-10 所示。

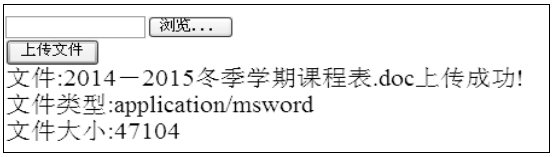


图 6-10　例 6-24 的运行界面

程序代码如下：

```
<%@ Page Language="C#" %>
<script runat="server">
protected void Button1_Click(object sender, EventArgs e)
{
    if (FileUpload1.HasFile)//判断 FileUpload 控件中是否有要上传的文件
        {
            string fileExtension=System.IO.Path.GetExtension(FileUpload1.FileName);
            //获取上传文件类型
            if (fileExtension != ".doc" && fileExtension != ".ppt" && fileExtension != ".xls")
            {
                Label1.Text ="文件类型错误，只能是：doc，ppt 或 xls！";
                return;
            }
            if (FileUpload1.PostedFile.ContentLength<6291456)//判断上传文件是否小于 6MB
            {
                try
                {
                    FileUpload1.PostedFile.SaveAs (Server.MapPath (".")+FileUpload1.FileName);
                        //上传文件至服务器当前目录
                    Label1.Text="文件:"+FileUpload1.FileName +"上传成功!"+"<br>";
                    Label1.Text=Label1.Text +"文件类型:
                        "+FileUpload1.PostedFile.ContentType+ "<br>";
```

```
                    Label1.Text=Label1.Text+"文件大小:
                        "+FileUpload1.PostedFile.ContentLength+ "<br>";
                }
                catch (Exception ex)
                {
                    Label1.Text="上传失败，重新上传!";
                }
            }
            else
            {
                Label1.Text="上传文件不能大于 6MB!";
            }
        }
        else
        {
            Label1.Text="未选择上传文件!";
        }
    }
    </script>
    <html>
    <head runat="server">
        <title>无标题页</title>
    </head>
    <body>
        <form id="form1" runat="server">
        <div>
            <asp:FileUpload ID="FileUpload1" runat="server" Height="20px" Width="189px" /> <br />
             <asp:Button ID="Button1" runat="server" Text="上传文件" onclick="Button1_Click"/><br />
             <asp:Label ID="Label1" runat="server" ></asp:Label>
        </div>
        </form>
    </body>
    </html>
```

6.4　ASP.NET 的对象

ASP.NET 中包含了许多开发者可利用的内置对象，提供基本的请求、响应、会话等处理功能，如 Response、Request 等。这些对象和服务器控件一样也是由.NET Framework 类来实现的，它们可以帮助用户高效地完成 Web 应用程序的开发。

6.4.1　对象简介

ASP.NET 定义的对象是在页面初始化请求时自动创建的，所以在程序中可以直接使用，不必事先声明。例如, 例 6-1 中的语句 Response.Write("欢迎你管理员同志 !"), 就是直接使用了 Response 对象。既然是对象，它们也和 Web 服务器控件一样具有对象的三要素：属性、事件和方法。

1．访问对象属性的语法格式

```
    对象名.属性名
```

例如：

```
    Request.Path;
```

是获取当前请求的物理路径。

2．访问对象方法的语法格式

```
对象名.方法名(参数表)
```

例如：

```
Response.Write("欢迎你管理员同志！");
```

输出欢迎词。

3．对象事件处理的定义语法格式

```
对象名_事件名(参数表) 或 事件名(参数表)
```

如例 6-20 中的 Page 对象事件：

```
protected void Page_Load(object sender, EventArgs e)
{
        if (!Page.IsPostBack)
        {
            DropDownList1.Items.Add("香蕉");
            DropDownList1.Items.Add("苹果");
            DropDownList1.Items.Add("梨子");
            DropDownList1.Items.Add("菠萝");
            DropDownList1.Items[0].Selected=true;
            }
}
```

在 ASP.NET 事件处理过程中都有以下两个参数：① object sender，表示发生该事件的源对象；② EventArgs e，表示传递给事件处理过程的额外描述，作为辅助之用。

ASP.NET 的常用对象如表 6-31 所示。

表 6-31　ASP.NET 的常用对象

对　象	功　能
Page	页面对象，用于整个页面的操作
Request	从客户端获取信息
Response	向客户端输出信息
Cookie	用于保存 Cookie
Session	存储特定用户的信息
Application	存储同一个应用程序中所有用户间的共享信息
Server	创建 COM 组件和进行有关设置
Mail	在线发送 E-mail

6.4.2　Page 对象

Page 对象用来设置与网页有关的各种属性、事件和方法，由命名空间 System.Web.UI 中的 Page 类来实现。Page 类用于表示一个.aspx 文件，又称为 Web 窗体页，因而 ASP.NET 的每个页面都派生自 Page 类，并继承这个类公开的所有方法和属性。

1．Page 对象的常用属性、方法和事件

Page 对象的常用属性如表 6-32 所示。

表 6-32　Page 对象的常用属性

属　性	说　明
ClientTarget	客户端浏览器属性
ErrorPage	当前网页发生未处理的异常时，将转向错误信息网页；若未设置此属性值，将显示默认错误信息网页
IsPostBack	网页加载状况。值为 False，表示网页第一次加载；值为 True，表示响应客户端请求而被重新加载。见例 6-20
IsValid	表示网页上的验证控件是否全部通过验证。值为 True，表示全部通过验证；值为 False，表示至少有一个验证失败
Visible	设置是否显示网页。True 显示，False 不显示

Page 对象的常用方法如表 6-33 所示。

表 6-33　Page 对象的常用方法

方　法	说　明
DataBind()	将数据源与页面上的服务器控件进行绑定
Dispose()	让服务器控件在释放内存前执行清理操作
FindControl(Id)	在页面上搜索标识为 Id 的服务器控件。若找到，返回该控件；若找不到，则返回 Nothing
HasControls()	Page 对象中包含服务器控件返回 True，否则返回 False
MapPath(VirtualPath)	将虚拟路径 VirtualPath 转换为实际路径

Page 对象的常用事件如表 6-34 所示。

表 6-34　Page 对象的常用事件

事 件 名	说　明
DataBinding()	当网页上的服务器控件连接数据源时触发该事件
Disposed()	网页从内存释放时触发该事件
Error()	网页上发生未处理的异常情况时触发该事件
Init()	当网页初始化时触发该事件。可以使用此事件来读取或初始化控件属性
Load()	当加载网页时触发该事件
Unload()	网页完成处理工作被卸载时触发该事件

Init()事件和 Load()事件的区别是：它们都是在加载网页时触发的事件，但 Init()事件先于 Load()事件，并且 Init()事件只在第一次加载时触发，所以只触发一次，而 Load()事件则可能触发多次。在例 6-20 中我们使用了 Page 对象的 IsPostBack 属性，如果将条件判断语句去掉，直接执行下拉框控件的 Add()方法就会发现，每单击一次命令按钮（“清除所有”按钮除外）就会重复添加下拉框控件中原来的选项。在这种情况下，必须将 Load()事件替换为 Init()事件，从而保证下拉框控件中的内容只添加一次。修改程序如下：

```
<script language="C#" runat="server">
    protected void Page_Init(object sender, EventArgs e)
     {
          DropDownList1.Items.Add("香蕉");
          DropDownList1.Items.Add("苹果");
          DropDownList1.Items.Add("梨子");
          DropDownList1.Items.Add("菠萝");
          DropDownList1.Items[0].Selected = true;
     }
</script>
```

2．Page 对象的使用

【例 6-25】Page 对象 FindControl()方法和 MapPath()方法的应用示例。执行以下程序，单击“提

交”命令按钮，显示结果如图 6-11 所示。

图 6-11　Page 对象的方法使用示例

程序代码如下：

```
<script runat="server">
void Button1_Click(object sender, EventArgs e)
{          Object a ;
           //搜索标识为 Button1 的服务器控件
           a = Page.FindControl("Button1");
           Response.Write(a);
           Response.Write("<br>");
          //将虚拟路径转换为物理路径
          Response.Write(Page.MapPath("VisualStudio2008"));
}
</script>
<html>
<head runat="server">
    <title>无标题页</title>
</head>
<body>
   <form id="form1" runat="server">
     <div>
     <asp:Button ID="Button1" runat="server" Text="提交" onclick="Button1_Click"/><br/>
     </div>
   </form>
</body>
</html>
```

6.4.3　Response 对象

1．Response 对象的常用属性和方法

Response 对象用于将服务器端的信息发送到客户端浏览器或重定向浏览器到另一个 URL。它派生自 HttpResponse 类，命名空间为 System.Web。

Response 对象的常用属性和方法，如表 6-35 和表 6-36 所示。

表 6-35　Response 对象的常用属性

属　性	说　明
BufferOutput	设置 HTTP 输出是否启用缓冲处理，默认为 True，启用
Charset	设置字符的编码方式
ContentType	获取或设置输出流的 HTTP MIME 类型
Cookies	设置客户端的 Cookies
Expires	获取或设置浏览器上缓存页过期之前的分钟数，如果用户在缓存之前返回同一页，则显示缓存的版本
IsClientConnected	获取客户端是否与服务器保持连接的信息

表 6-36　Response 对象的常用方法

方　法	说　明
ClearContent()	清除缓冲区的内容
End()	输出当前缓冲区的内容并终止当前页面的处理
Flush()	将当前缓冲区的内容发送到客户端并清除缓冲区
Redirect()	将客户端重定向到新的 URL
Write()	将指定的内容写入页面文件
WriteFile(filename)	将指定文件的内容直接输出至客户端

2．Response 对象的使用

（1）输出文本

Write()是 Response 对象中最常用的一个方法，它可以把信息从服务器端直接发送到客户端。使用语法如下：

```
Response.Write( String )
```

其中，String 为变量或字符串。字符串可以是 HTML 标记，发送到客户端由浏览器解释执行。例如：

```
Response.Write("欢迎"+"<font size=15>"+"使用 ASP.NET" + "<br>");
```

【例 6-26】使用 Write()方法输出信息。运行结果如图 6-12 所示。

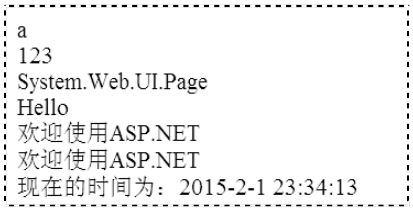


图 6-12　例 6-26 的运行结果

程序代码如下：

```
<script runat="server">
    protected void Page_Load(object sender, EventArgs e)
    {
        char c = 'a';//定义一个字符变量
        int digit = 123;//定义一个整型变量
        Page p = new Page();//定义一个 Page 对象
        char[] carray = {'H','e','l','l','o' };//定义一个字符数组
        string str = "欢迎使用 ASP.NET" + "<br>";//定义一个字符串变量
        Response.Write(c);//输出字符变量的值
        Response.Write("<br>");//输出 HTML 标记
        Response.Write(digit);//输出数据变量的值
        Response.Write("<br>");//输出 HTML 标记
        Response.Write(p); //输出对象
        Response.Write("<br>");//输出 HTML 标记
        Response.Write(carray,0,carray.Length) ;//输出字符数组
        Response.Write("<br>");    //输出 HTML 标记
        Response.Write(str);//输出字符串变量的值
        Response.Write("欢迎使用 ASP.NET" + "<br>"); //输出字符串
        Response.Write("现在的时间为：" + DateTime.Now); //输出函数值
    }
    </script>
```

由运行结果可以看出，输出"欢迎使用 ASP.NET"的两种方法是完全等价的。

（2）网页重定向

与 Write()方法将服务器端的信息送到客户端不同，Redirect()方法引导客户端的浏览器立即重定向到程序指定的 URL 位置，也就是进入到另一个 Web 页面，类似于 HTML 文件中的超链接。其使用语法如下：

```
Response.Redirect（String）;
```

其中，String 值可以是一个确定的 URL，或是一个虚拟目录下的文件名，还可以是一个与请求页面存储在同一个文件夹下的文件名。如：

```
Response.Redirect("other.aspx");//跳转到当前目录的另一个页面
Response.Redirect("~/index.aspx");//跳转到根目录下的 index.aspx 页面
```

【例 6-27】使用 Redirect()方法重定向到另一个 URL。设计一个程序将搜索引擎 Baidu 嵌入到自己的网页中。程序界面如图 6-13 所示。

图 6-13　重定向到搜索页面

程序代码如下：

```
<script runat="server">
void Button1_Click(object sender, EventArgs e)
{
    if (TextBox1.Text != "" )
       Response.Redirect("http://www.baidu.com/s?wd=" +TextBox1.Text);
}
</script>
<html>
<head runat="server">
    <title>无标题页</title>
</head>
<body>
   <form id="form1" runat="server">
    <div>
      请输入<asp:TextBox ID="TextBox1" runat="server"></asp:TextBox>
          <asp:Button ID="Button1" runat="server" Text="搜索" onclick="Button1_Click"/>
    </div>
   </form>
</body>
</html>
```

运行程序，在文本框中输入“ ASP.NET ”字符串，单击“ 搜索 ”按钮，立即进入 baidu 搜索引擎，查找与 ASP.NET 有关的资料。如果将程序中的重定向语句改成：

```
Response.Redirect("http://www.baidu.com/");
```

程序运行时将直接跳转到百度主页。

（3）输出文本文件

一般来说，少量数据使用 Write()方法输出。当有大量数据要发送到浏览器时，使用 WriteFile()方法，可以提高程序的可读性，因为将要输出的数据是存放在数据文件中的。使用语法如下：

```
Response. WriteFile（filename）
```

【例 6-28】利用 WriteFile()方法将文本文件 info.txt 的内容发送到浏览器。程序代码如下：

```
<script runat=server>
  protected void Page_Load(object sender, EventArgs e)
{
        Response.Charset = "gb2312" ;          //设置字符编码方式
        Response.WriteFile("Info.txt");        //当前目录下的文件
```

```
    }
    </script>
```

这里假定文本文件 Info.txt 与网页文件在同一目录下。若不在同一目录下，则需指明文件所在的位置。例如：

```
Response.WriteFile("..//Info.txt");   //父目录下的文件
Response.WriteFile("\\obj\\Info.txt"); //子目录 obj 下的文件
Response.WriteFile("d:\\myweb\\Info.txt");   //绝对路径的文件
```

（4）缓冲处理

当 BufferOutput 属性值设置为 True 时，输出信息暂时存放在服务器的缓冲区，当程序结束或接受到 Flush()或 End()命令后，才会将信息从缓存发出。

【例 6-29】BufferOutput 属性和 Flush()、ClearContent()方法的使用。运行结果如图 6-14 所示。

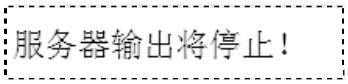


图 6-14　例 6-29 的运行结果

程序代码如下：

```
<%@ Page Language="C#" %>
<script runat="server">
protected void Page_Load(object sender, EventArgs e)
{
    Response.BufferOutput = true;//设置服务器缓冲为 true
}
protected void Button1_Click(object sender, EventArgs e)
{
    Response.Write("服务器输出将停止！");//发送信息到浏览器
    Response. Flush ();//立即发送缓冲区的数据
    Response.Write("服务器输出已停止！");//这个信息没有输出到浏览器
    Response.ClearContent();//清除缓冲区中的数据
    Response.End ();//停止执行代码
}
</script>
<html>
<head runat="server">
    <title>无标题页</title>
</head>
<body>
    <form id="form1" runat="server">
    <asp:Button ID="Button1" runat="server" Text="停止服务器输出" onclick="Button1_Click"/>
    </form>
</body>
</html>
```

由于设置了服务器缓存，需要输出的信息并不是立即发送到浏览器。当按下“停止服务器输出”按钮，执行到 Flush()命令，才将信息发出，浏览器显示“服务器输出将停止！”的提示信息。接下来缓存的“服务器输出已停止！”信息，因为 ClearContent()指令又被清除了，所以浏览器上不会看到此信息。

【例 6-30】使指定的网页在上午 8:00~10:00 显示，其余时间若有用户来访，则给出提示信息并结束程序的执行，不显示网页内容。程序界面如图 6-15 所示。

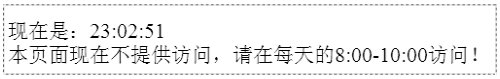


图 6-15　例 6-30 的运行结果

程序代码如下：

```
<%@ Page Language="C#" %>
<script runat="server">
protected void Page_Load(object sender, EventArgs e)
{
    int currtime;
    currtime = DateTime.Now.Hour;
    if (currtime > 10 || currtime < 8)
    {
        Response.Write("现在是：" + DateTime.Now.ToLongTimeString()+"<br>");
        Response.Write("本页面现在不提供访问，请在每天的 8:00-10:00 访问！");
        Response.End();
    }
}
</script>
<html>
<head runat="server">
    <title>无标题页</title>
</head>
<body>
    <p>  欢迎访问！</p>
</body>
</html>
```

若用户在 8:00~10:00 点访问该网页，显示“ 欢迎访问 !”。其他时间，则显示如图 6-15 所示的提示信息。因为程序中使用了 Response.End()方法结束程序的执行，所以“ 欢迎访问 !” 的信息不再输出到客户端。

6.4.4 Request 对象

Request 对象派生自 HttpRequest 类，命名空间为 System.Web。它的主要作用是从客户端获取数据，包括使用 Post()方法和 Get()方法传递参数等。

Request 对象的使用语法如下：

```
Request{. 数据集合 | 属性 | 方法}[参数]
```

其中，符号[ ]表示这个参数是可以省略的。

1．Request 对象的常用属性和方法

Request 对象的常用属性如表 6-37 所示。Request 对象还包含多个有用的数据集合，它们是只读的，表 6-38 列出了 Request 对象常用的数据集合。Request 对象的常用方法，如表 6-39 所示。

表 6-37　Request 对象的常用属性

属　　性	说　　明
ApplicationPath	获取 ASP.NET 应用的虚拟目录（URL）
HttpMethod	获取客户端使用的 HTTP 数据传输方式（Post 或 Get 或 Head）
Path	获取当前请求网页的虚拟路径和网页名称
TotalBytes	获取客户端请求数据的字节数
PhysicalPath	获取当前请求网页的物理路径
Url	获取当前请求网页的 URL 完整信息
UserHostAddress	获取客户端的 IP 地址
UserHostName	获取客户端的 DNS 名称

表 6-38　Request 对象的常用数据集合

集　　合	说　　明
Cookies	获取客户端的 Cookies 信息
Browser	获取客户端浏览器支持的功能信息
Form	获取客户端表单元素中所填入的信息
QueryString	获取当前请求 URL 上的附加数据
ServerVariable	获取服务器端环境变量的值，它由一些预定义的服务器环境变量组成

表 6-39　Request 对象的常用方法

方　　法	说　　明
Mappath(VirtualPath)	将参数 VirtualPath 指定的虚拟路径映射为实际路径
SaveAs(filename,includeHeaders)	将 HTTP 请求保存到磁盘。其中参数 filename 是保存文件的路径及文件名，includeHeaders 指定是否保存 HTTP 标头

2．Request 对象的使用

（1）获取客户端信息

使用 Request 对象的 Browser 数据集合可以方便的获取客户端浏览器的特性。常见的浏览器特性如表 6-40 所示。

表 6-40　常见浏览器特性

名　　称	说　　明
ActiveXcontrols	是否支持 ActiveX 控件
BackgroundSounds	是否支持背景音乐
Browser.	浏览器类型名称
Cookies	是否支持 Cookies
Frames	是否支持框架
JavaApplets	是否支持 Java 小程序
JavaScript	是否支持 JavaScript
Platform	客户端操作系统版本
Tables	是否支持表格
VBScript	是否支持 VBScript
Version	浏览器版本名称

【例 6-31】使用 Request 对象的属性和 Browser 数据集合，设计一个获取客户端机器和浏览器信息的页面。运行结果如图 6-16 所示。

浏览器版本：8.0
浏览器名称：IE
客户端操作系统：WinXP
客户端IP地址：127.0.0.1
当前请求的虚拟路径：/Default6.aspx
当前请求的物理路径：d:\我的文档\桌面\新建文件夹 (2)\WebApplication1\Default6.aspx

图 6-16　例 6-31 的运行结果

程序代码如下：

```
<script runat=server>
        protected void Page_Load(object sender, EventArgs e)
{
            Response.Write("浏览器版本："+ Request.Browser.Version + "<br>");
            Response.Write("浏览器名称：" + Request.Browser.Browser + "<br>");
```

```
            Response.Write("客户端操作系统：" +Request.Browser.Platform + "<br>");
            Response.Write("客户端 IP 地址：" + Request.UserHostAddress +"<br>");
            Response.Write("当前请求的虚拟路径：" + Request.Path + "<br>");
            Response.Write("当前请求的物理路径：" + Request.PhysicalPath + "<br>")
    }
    </script>
```

（2）获取表单数据

表单是标准 HTML 语言的一部分，它允许用户利用表单中的文本框、复选框、单选钮、列表框等控件为服务器端的应用提供初始数据，用户通过单击表单中的命令按钮提交他们的输入数据。服务器获取表单数据的方式取决于客户端提交的方式。

若提交方式为“ post ”，使用 Request 对象的 Form 集合来获取客户端的信息。其使用语法如下：

```
Request.Form [ String 参数]
```

其中，String 参数指定集合要检索的表单元素名称（如 Text、Radio 等）。

若提交方式为“ get ”，使用 Request 对象的 QueryString 数据集合来获取信息。其使用语法如下：

```
Request.QueryString [String 参数]
```

【例 6-32】使用 Request 对象的 Form 集合在网页之间传递数据。本例在页面 6-32-1.aspx 中输入用户名和密码，提交给页面 6-32-2.aspx，页面 6-32-2.aspx 通过 Form 集合获取并显示提交的信息。运行界面分别如图 6-17 和图 6-18 所示。

图 6-17　程序 6-32-1.aspx 的运行界面

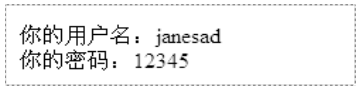


图 6-18　程序 6-32-2.aspx 的运行界面

程序 6-32-1.aspx 的代码如下：

```
<html><head id="Head1" runat="server"> <title>无标题页</title></head>
<body>
<form id="form1" method="post" action="6-32-2.aspx">
    用户名<input name="usename" type="text" /><br />
    密码<input name="usepass" type="password" /><br />
    <input type="submit" value="提交" />
</form>
</body>
</html>
```

程序 6-32-2.aspx 的代码如下：

```
<script runat="server">
  protected void Page_Load(object sender, EventArgs e)
  {
            Response.Write("你的用户名：" + Request.Form["usename"] + "<br>");
            Response.Write("你的密码：" + Request.Form["usepass"] + "<br>");
  }
  </script>
```

【例 6-33】 使用 Request 对象的 QueryString 集合在网页之间传递数据。将程序 6-32-1.aspx 中 Form 的 method 属性值修改为 get。将程序 6-32-2.aspx 的代码修改为如下形式：

```
<script runat="server">
protected void Page_Load(object sender, EventArgs e)
```

```
{
    Response.Write("你的用户名：" + Request.QueryString["usename"] + "<br>");
    Response.Write("你的密码：" + Request.QueryString["usepass"] + "<br>");
}
</script>
```

再次运行程序 6-33-1.aspx，可以得到与图 6-17 和图 6-18 同样的界面。

比较上述两种在网页之间传递数据的方法可以看到，除了实现方式上有所区别，在传递数据上，Form 集合方式的地址栏里不带有传递数据的参数值，如下所示：

```
http://localhost/WebSite1/6-32-2.aspx
```

而 QueryString 集合方式的地址栏里带有传递数据的参数值，如下所示：

```
http://localhost/WebSite1/6-33-2.aspx?usename=janesad&usepass=12345
```

由此可见，使用 QueryString 集合传递数据是不安全的。事实上，Form 集合现在已很少使用，更多的是使用 Web 服务器控件来制作表单，这样就可以直接访问服务器控件，从而得到用户的输入信息。

（3）获取服务器端环境变量值

ServerVariables 数据集合帮助客户取得服务器端的环境变量信息。它由一些预定义的服务器环境变量组成，这些变量是只读的，只能查阅，不能设置。表 6-41 列出了一些主要的服务器环境变量。其使用语法如下：

```
Request.ServerVariables[服务器环境变量];
```

表 6-41　服务器环境变量

名　称	说　明
ALL_HTTP	客户端发送的所有 HTTP 标题文件
CONTENT_LENGTH	客户端提交的正文长度
CONTENT_TYPE	正文的数据类型，如 text/html
LOCAL_ADDR	返回接受请求的服务器地址。在绑定多个 IP 地址的多宿主机器上查找请求所使用的地址时，这一变量非常重要
LOGON_USER	用户登录 Windows NT 的账号
PATH_TRANSLATED	获取当前页面的物理路径
QUERY_STRING	查询 HTTP 请求中问号（?）后的信息
REMOTE_ADDR	发出请求的远程主机的 IP 地址
REMOTE_HOST	发出请求的主机名称
REQUEST_METHOD	获取表单提交正文的方法，如 GET、POST 等
SERVER_NAME	获取服务器的主机名、DNS 域名或 IP 地址
SERVER_PORT	服务器端连接的端口号
SCRIPT_NAME	服务器端脚本文件的虚拟路径

【例 6-34】通过 ServerVariable 集合获取服务器端环境变量。如服务器名称、服务器 IP 地址、服务器连接端口等。程序运行结果如图 6-19 所示。

图 6-19　获取的服务器端环境变量的值

程序代码如下：

```
<script runat="server">
  protected void Page_Load(object sender, EventArgs e)
{
      Response.Write("服务器名称:" + Request.ServerVariables["server_name"] + "<br>");
      Response.Write("服务器连接端口:" + Request.ServerVariables["server_port"] + "<br>");
      Response.Write("服务器端的 IP 地址:" + Request.ServerVariables["local_addr"] + "<br>");
      Response.Write("当前网页的实际路径:" +
      Request.ServerVariables["path_translated"] + "<br>");
}
</script>
```

6.4.5 Application 对象

1．Application 对象的常用属性、方法和事件

Application 对象派生自 HttpApplicationState 类，用于保存所有客户的公共信息，在第一个用户请求 ASP.NET 文件时自动创建。一旦 Application 对象被创建，它就可以共享和管理整个应用程序的信息，并在服务器运行期间持久地保存。Application 对象的生命周期起始于 Web 服务器开始执行，终止于 Web 服务器关机或重新启动。

利用 Application 特性，可以创建“聊天室”和“网站计数器”等网页应用程序。Application 对象的常用属性如表 6-42 所示。Application 对象的常用方法如表 6-43 所示。

表 6-42　Application 对象的常用属性

属　性	说　明
AllKeys	AllKey 返回所有的变量名，AllKeys(index)返回下标为 index 的变量名
Count	获取 Application 对象变量的个数
Contents	获取对 Application 对象的引用

表 6-43　Application 对象的常用方法

方　法	说　明
Add(name,value)	添加一个新的 Application 对象变量，名为 name，值为 value
Clear()	清除所有 Application 对象变量
Get({name,index})	获取名称为 name 或下标为 index 的变量值
GetKey(index)	获取索引值为 index 的变量名
Lock()	锁定。禁止其他用户修改 Application 对象变量
Remove(name)	清除名为 name 的变量
RemoveAll()	清除所有的 Application 对象变量
RemoveAt(index)	清除索引为 index 的变量
Set(name,value)	将名为 name 的变量值修改为 value
UnLock()	解除对 Application 对象变量的锁定

Application 对象事件主要有以下 4 个。

（1）Start()事件：在整个 ASP.NET 应用程序第一次执行时触发。

（2）End()事件：与 OnStart()事件正好相反，在整个应用程序结束时触发。

（3）BeginRequest()事件：在每一个 ASP.NET 被请求时触发。客户每访问一次 ASP.NET 程序，就触发一次该事件。

（4）EndRequest()事件：结束 ASP.NET 程序时，触发该事件。

Application 对象的事件过程定义格式如下：

```
protected void Application_事件名(object sender, EventArgs e)
{
…
}
```

这些事件过程的代码由用户编写，保存在 Global.asax 文件中，并存放在网站的根目录下。当客户端发出网页请求时，首先检查根目录下是否有 Global.asax 文件，如果有就先执行该文件中的相关事件过程。

2．Application 对象的使用

（1）存、取信息

使用 Application 对象存取信息，就需要定义 Application 变量。Application 对象定义的变量为应用程序级变量。变量可以在 Global.asax 文件或 aspx 页面中声明。其使用语法如下：

```
Application["变量名"]=变量|常量|字符串表达式；
```

例如：

```
Application["MyVar"] = "Hello";//定义 Application 变量
Response.Write(Application["MyVar"]); //读取 Application 变量信息并输出
```

一旦定义了 Application 变量，它将持久地保存在服务器内存中，直到服务器关闭，除非使用 Remove()方法将其删除。例如：

```
Application. Remove["MyVar"];
```

（2）加锁与解锁

Application 变量的数据被整个应用程序所共享，即所有用户都可对其修改和存取。为了保证数据的一致性和完整性，ASP.NET 引入了加、解锁机制。使用前通过 lock()方法对 Application 对象加锁，使用后调用 Unlock()方法解锁，确保同一时刻只有一个用户对 Application 变量进行操作。

【例 6-35】 使用 Application 对象和 RadioButtonList 控件设计一个 Web 应用程序，统计网民的学历分布情况。可分为：博士、硕士、本科、大专、高中、初中 6 个层次。运行界面如图 6-20 所示。程序代码分为两部分，一部分存放在文件 Global.asax 中，另一部分存放在扩展名为.aspx 的文件中。

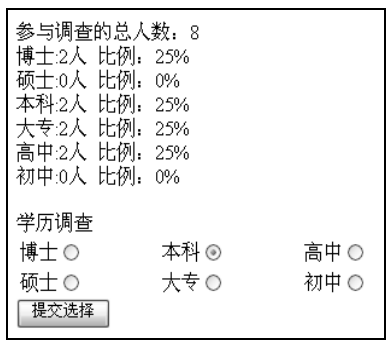


图 6-20　网民学历调查值

Global.asax 文件中的内容：

```
<script runat="server">
    protected void Application_Start(object sender, EventArgs e)
    {
            int []count=new int[6];
             //定义一个数组型的 Application 对象变量
            Application["storecount"] = count;          //A 句
            //定义一个统计调查人数的 Application 对象变量
            Application["users"]=0;
     }
   protected void Application_End(object sender, EventArgs e)
   {
             //应用结束时，清除所有 Application 对象变量
            Application.RemoveAll();
   }
</script>
```

扩展名为.aspx 的文件内容：

```
<script   runat=server>
protected void Button1_Click(object sender, EventArgs e)
{
    int i;
     //定义一个数组，用来统计各学历层次的人数
    int []count=(int[])Application["storecount"];//用 Application 对象对其初始化
        //查询选中项，并将其对应的数组值加 1
    for(i = 0 ;i<=5;i++)
          if (RadioButtonList1.Items[i].Selected == true )
              count[RadioButtonList1.SelectedIndex]= count[RadioButtonList1.SelectedIndex] + 1;
    Application.Lock();     //加锁
      //将参与调查的人数加 1
    Application["users"] = (int)Application["users"] + 1;         //B 句
      //保存统计结果到 Application 对象变量中
    Application["storecount"] = count;
    Application.UnLock();     //解锁
      //输出调查结果
    Response.Write("参与调查的总人数：" + Application["users"] + "<br/>");
    for(i = 0;i<=5;i++)
    {
        Response.Write(RadioButtonList1.Items[i].Text + ":");
        Response.Write(count[i] + "人  比例："
         + count[i] * 100.0f/(int) Application["users"]    + "%<br/>");
    }
}
</script>
<html><head runat="server"> <title>无标题页</title></head>
<body>
    <form id="form1" runat="server" >
          学历调查<br />
     <asp:RadioButtonList ID="RadioButtonList1" runat="server" RepeatColumns="3" TextAlign=
        "Left" Width="356px">
                    <asp:ListItem Value="1">博士</asp:ListItem>
                    <asp:ListItem Value="2">硕士</asp:ListItem>
                    <asp:ListItem Value="3">本科</asp:ListItem>
                    <asp:ListItem Value="4">大专</asp:ListItem>
                    <asp:ListItem Value="5">高中</asp:ListItem>
                    <asp:ListItem Value="6">初中</asp:ListItem>
     </asp:RadioButtonList>
     <asp:Button ID="Button1" runat="server" Text="提交选择" onclick="Button1_Click" />
     </form>
</body>
</html>
```

Global.asax 文件中，包含两个事件过程。在 Application_Start()事件过程中，创建了一个数组型的 Application 对象变量 storecount，存放各学历层次的人数，其中 A 句可替换为 Application.Add ("storecount", count)。另外还创建了一个统计参与人数的 Application 对象变量 users。在 Application_ End()事件过程中, 调用 Application 对象的 RemoveAll()方法, 清除所用的 Application 对象变量。.aspx 文件中，Button1_Click()事件过程中的 B 句也可以用 Application.Set("users", Application("users") + 1) 替换。Application.Lock()方法的使用是为了避免多个用户同时修改 Application 变量的值。加锁以后，

同一时刻只允许一个用户对其进行操作，直到调用 Application.UnLock()方法解锁为止。

需要说明的是，该程序存在严重漏洞。当用户在同一个浏览器中反复选择，多次单击“ 提交 ”按钮时，统计数字会不断更新，从而造成调查数据的不可靠。

6.4.6　Session 对象

Session 对象派生自 System.Web.SessionState 类，命名空间为 System.Web。Session 对象记录特定用户的信息，即使客户从一个页面跳转到另一个页面，该 Session 信息仍然存在，客户在该网站的任何一个页面都可以存取 Session 信息。它与 Application 对象的区别在于 Session 对象是每个连接用户独自拥有，而 Application 对象是所有连接用户共同拥有。也就是说，某一个时刻若有 10 个连接者，则 Session 对象个数为 10，而 Application 对象个数为 1。

Session 对象通常用来记录单个用户的信息，如身份密码、个人喜好等。

1．Session 对象的常用属性、方法和事件

Session 对象的常用属性如表 6-44 所示。Session 对象的常用方法如表 6-45 所示。

表 6-44　Session 对象的常用属性

属　性	说　明
Count	获取 Session 对象变量的个数
IsReadOnly	该值指示会话是否为只读，默认值为 False
IsNewSession	该值指示会话是否与当前请求一起被创建
SessionID	用于标识每个 Session 对象的标识码
Timeout	设置 Session 对象的失效时间，单位为分钟，默认为 20 分钟

表 6-45　Session 对象的常用方法

方　法	说　明
Add(name，value)	增加一个新的 Session 变量，名为 name，值为 value
Clear()	清除全部 Session 变量的值
Abandon()	释放 Session 对象，调用此方法将触发 OnEnd 事件
Remove(name)	清除名称为 name 的 Session 变量
RemoveAll()	清除所有的 Session 变量
RemoveAt(index)	清除会话集合中下标为 index 的 Session 变量

与 Application 对象相同，Session 对象的事件过程代码也是由用户编写，保存在 global.asax 文件中。Session 对象有以下两个主要事件。

（1）Start()事件。当用户首次请求 ASP.NET 网页时，将创建 Session 对象并触发该事件。同一浏览器只会触发此事件一次，除非发生 End()事件或重新启动浏览器。

（2）End()事件。在用户执行 Session.Abandon()方法或在 Timeout 属性设置的时间之内没有再次访问网页时都会触发该事件。用户在客户端直接关闭浏览器，并不会触发该事件。该事件通常用来处理用户结束会话后的善后工作，如保存相关数据或改写在线人数等。

Session 对象的事件过程定义格式如下：

```
protected void Session_事件名(object sender, EventArgs e)
{
//在新会话启动时运行的代码
}
```

2．Session 对象的使用

（1）存、取信息

Session 对象定义的变量称为会话变量。每个访问用户单独拥有一个 Session 变量，存储用户会话所需的信息，其他用户不能访问和修改这个变量。因此，读写 Session 变量时，不需要任何加、解锁机制。Session 对象定义变量的方法与 Application 对象相同，在第一次给 Session 变量赋值时自动创建。语法格式如下：

```
Session ["变量名"]=变量|常量|字符串表达式；
```

例如：

```
Session ["MyVar"] = "World"; //定义 Session 变量
Response.Write(Session ["MyVar"]); //读取 Session 变量信息
```

（2）设置页面有效期

每一个 Session 对象都有自己的有效期。如果用户在指定时间内没有请求或刷新应用程序中的任何页，会话将自动结束并释放出占用的资源，即删除 Session 变量。这段时间的默认值是 20 min(分钟)。若想改变这个值，可修改 Session 对象的 Timeout 属性。当然也可通过 Session.Abandon()方法强制释放 Session 对象占用的资源。

【例 6-36】利用 Session 对象和 Application 对象设计一个用户登录页面，要求在另一个页面中显示用户名、网站的点击率和网站的在线人数。若跳过登录页面，则强制转向登录页面。程序代码分别存放在 Global.asax、6-36-1.aspx、6-36-2.aspx 三个文件中。

Global.asax 文件的内容：

```
<script runat="server">
    protected void Application_Start(object sender, EventArgs e)
    {
        //定义一个 Application 对象变量,记录网站的访客数
        Application["usercount"] = 0;
        //定义一个 Application 对象变量,记录网站的在线人数
        Application["onlinecount"] = 0;
    }
    protected void Application_End(object sender, EventArgs e)
    {
        //应用结束时，清除所有 Application 对象变量
        Application.RemoveAll();
    }
    protected void Session_Start(object sender, EventArgs e)
    {
        Application.Lock();     //加锁
//开始一个新的会话，访问计数加 1
        Application["usercount"] =(int)Application["usercount"] + 1;
        //开始一个新的会话，在线人数加 1
        Application["onlinecount"] = (int)Application["onlinecount"] + 1;
        Application.UnLock();     //解锁
Session["username"] = "";
        Session.Timeout = 1; //会话有效期设置为 1，方便观察
    }
    protected void Session_End(object sender, EventArgs e)
    {
        Application.Lock();     //加锁
//结束一个会话，在线人数减 1
        Application["onlinecount"] = (int)Application["onlinecount"] - 1;
```

```
                Application.UnLock();     //解锁
        }
    </script>
```

登录页面文件 6-36-1.aspx 的内容：

```
    <script runat="server">
        //登录按钮的单击事件过程
      protected void Button1_Click(object sender, EventArgs e)
    {          if( usename.Text != "")
               {
                  Session["username"] = usename.Text; //输入的用户名存入 Session 变量中
                  Session["loginID"] = true;   //设置登录标志变量
                  Response.Redirect("6-36-2.aspx");   //将网页转向显示页面
               }
      }
    </script>
    <html><head runat="server"> <title>无标题页</title></head>
    <body>
        用户登录<hr/>
    <form id="form1" runat="server" >
          用户名：<asp:TextBox ID="usename" runat="server"></asp:TextBox><br/>
                <asp:Button ID="Button1" runat="server" Text="登录"
              onclick="Button1_Click" />
    </form>
    </body>
    </html>
```

显示页面文件 6-36-2.aspx 的内容：

```
    <script   runat=server>
    protected void Page_Load(object sender, EventArgs e)
    {
    if ((bool)Session["loginid"]==true)       //测试是否经过登录页面
        { //在显示页面输出 Session["username"]变量的值
            Response.Write(Session["username"] + "你好！" + "<br> ");
            Response.Write("你是第" +Application["usercount"] + "位访客" + "<br>");
            Response.Write("目前在线人数:" + Application["onlinecount"] + "<br>");
            Session["loginid"] = false;
        }
        else
            Response.Redirect("6-36-1.aspx"); //将网页转向登录页面
    }
    </script>
```

运行程序 6-36-1.aspx，看到如图 6-21 所示的登录界面。输入用户名，单击“登录”按钮，转向图 6-22 所示的显示界面。若是直接运行 6-36-2.aspx 程序，则会跳转到登录页面。因为在登录页面中设置了一个登录标志变量 Session["loginID"]，只有经过登录页面该变量的值才会置为 true，从而通过显示页面的测试。

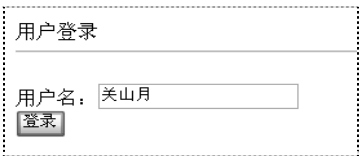


图 6-21　登录界面

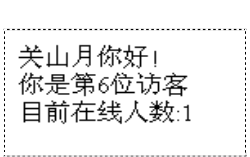


图 6-22　显示界面

【例 6-37】修改例 6-35 程序，防止用户在一个浏览器中多次提交信息。

在 Global.asax 文件中增加一个 Session_start()事件过程。程序代码如下：

```
protected void Session_Start(object sender, EventArgs e)
{
        //建立首次提交标志 Session 变量 isfirst，并将其值设为 true
        Session["isfirst"] = true;
}
```

修改.aspx 文件代码声明块如下：

```
<script   runat=server>
protected void Button1_Click(object sender, EventArgs e)
{
    int i;
     //定义一个数组，用来统计各学历层次的人数
    int []count=(int[])Application["storecount"];//用 Application 对象对其初始化
        //查询选中项，并将其对应的数组值加 1
    for(i = 0 ;i<=5;i++)
          if (RadioButtonList1.Items[i].Selected == true )
              count[RadioButtonList1.SelectedIndex]= count[RadioButtonList1.SelectedIndex] + 1;
    if ((bool)Session["isfirst"] == true)     //首次提交，才是有效信息
    {
        Application.Lock();     //加锁
        //将参与调查的人数加 1
        Application["users"] = (int)Application["users"] + 1;         //B 句
        //保存统计结果到 Application 对象变量中
        Application["storecount"] = count;
        Application.UnLock();     //解锁
        //输出调查结果
        Response.Write("参与调查的总人数：" + Application["users"] + "<br/>");
        for (i = 0; i <= 5; i++)
        {
            Response.Write(RadioButtonList1.Items[i].Text + ":");
            Response.Write(count[i] + "人  比例："
              + count[i] * 100.0f / (int)Application["users"] + "%<br/>");
        }
        Session["isfirst"] = false; //将首次提交标志变量设置为 false
    }
    else
        Response.Write("你已经参与过统计！不可再参加！");
}
</script>
```

程序中的粗体代码为修改后增加的内容。

6.4.7 Server 对象

Server 对象派生自 HttpServerUtility 类，命名空间为 System.Web。Server 对象是专为处理服务器上的特定任务而设计的，特别是与服务器的环境和处理活动有关的任务。

1．Server 对象的常用属性和方法

Server 对象的常用属性如表 6-46 所示。

表 6-46　Server 对象的常用属性

属　性	说　明
MachineName	获取服务器端机器的名称，是只读属性
ScriptTimeout	获取和设置程序执行的最长时间，即程序必须在该时间段内执行完毕。单位为秒，系统默认值为 90 秒

例如，使用 Server 对象的 MachineName 属性获取计算机名称：

```
Response.Write(Server.MachineName)
```

Server 对象的常用方法如表 6-47 所示。

表 6-47　Server 对象的常用方法

方　法	说　明
CreateObject(type)	创建 COM 对象的一个服务器实例
Execute(path)	执行由 path 指定的 ASP.NET 程序，执行完毕后仍继续原程序的执行
HtmlEncode(string)	对要在浏览器中显示的字符串 string 进行编码
HtmlDecode(string)	与 HtmlEncode 相反，还原为原来的字符串
MapPath(path)	将参数 path 指定的虚拟路径转换为物理路径
Transfer(path)	结束当前 ASP.NET 程序的执行，并开始执行参数 path 指定的程序
UrlEncode(string)	对字符串 string 以 Url 格式进行编码
UrlDecode(string)	对 Url 格式字符串进行解码

2．Server 对象的使用

（1）HtmlEncode()和 HtmlDecode()方法

普通的输出语句中如果包含 HTML 标签，将会被浏览器解释为 HTML 的内容，得不到所希望的结果。如果使用的 Server 对象的 HtmlEncode()方法就可将 HTML 标记原样输出。而 HtmlDecode()方法，则可将 HTML 编码字符按 HTML 语法进行解释。

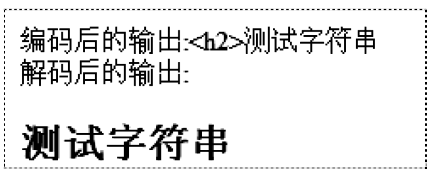


图 6-23　对 HTML 标记的编码和解码

【例 6-38】HtmlEncode()和 HtmlDecode()方法应用示例。程序的运行结果如图 6-23 所示。

程序代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
        string str;
        str = "<h2>测试字符串";
        Response.Write("编码后的输出:" +Server.HtmlEncode(str));
        Response.Write("<br>");
        Response.Write("解码后的输出:" + Server.HtmlDecode(str));
 }
```

（2）UrlEncode()和 UrlDecode()方法

在利用 Request 对象的 QueryString 集合获取标识在 URL 后面的参数时，参数可能带有空格、“ ？ ”、中文等特殊字符，如"login.aspx?name=关山月"，如何处理呢？此时，这些特殊字符都被进行了 URL 编码，从而保证了浏览器中提交的文本能够正确传输。

【例 6-39】UrlEncode()和 UrlDecode()方法应用示例。程序运行结果如图 6-24 所示。

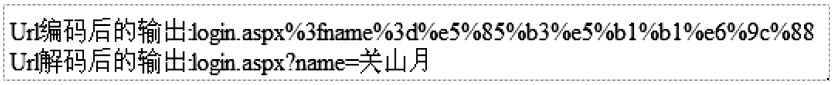


图 6-24　URL 方法对字符串的编码和解码

程序代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
        String str;
        str = "login.aspx?name=关山月";
        Response.Write("编码后的输出:" + Server.UrlEncode(str));
        Response.Write("<br>");
        Response.Write("解码后的输出:" + Server.UrlDecode(str));
 }
```

（3）利用 Server 对象进行路径映射

在应用程序中给出的文件路径通常都是虚拟路径，若想将虚拟路径转换为实际的物理路径或想得到服务器的根目录，可使用 Server 对象的 MapPath()方法。例如：

```
Server.MapPath(".");              //获得当前目录的物理路径
Server.MapPath("/");              //获得服务器根目录的物理路径
```

（4）执行指定程序

前面介绍过使用 Response 对象的 Redirect()方法，可以转向新的网页去执行。而 Server 对象的 Execute()方法和 Transfer()方法也可以转向新的网页。与 Response 对象的重定向发生在客户端不同，Server 对象的重定向发生在服务器端，并且 Server 对象只能重定向到同一个应用中的其他文件，而 Response 对象可以重定向到其他网站。Execute()方法和 Transfer()方法的区别在于：Execute()方法执行新的网页后，仍然返回原网页继续执行，类似于其他语言中的函数和过程调用；而 Transfer()方法执行完新的网页后，则停止运行。

【例 6-40】Execute()方法和 Transfer()方法比较应用示例。

程序 6-40-1.aspx 的代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
        Response.Write("Execute 方法的使用" + "<br>");
        Server.Execute("6-40-2.aspx");
        Response.Write("Transfer 方法的使用" + "<br>");
        Server.Transfer("6-40-2.aspx");
        Response.Write("程序结束");
}
```

程序 6-40-2.aspx 的代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
        Response.Write("Execute 方法和 Transfer 方法比较" + "<br>");
}
```

运行程序 6-40-1.aspx，结果如图 6-25 所示。

由图 6-25 不难看出，执行 Server.Execute ("6-40-2.aspx")语句，转向程序 6-40-2.aspx，执行完毕后，仍返回程序 6-40-1.aspx，继续执行 Server.Transfer("6-40-2.aspx")语句，此时再次转向程序 6-40-2. aspx，运行完毕没有返回程序 6-40-1.aspx，所以图中没有看见“程序结束”的输出显示。

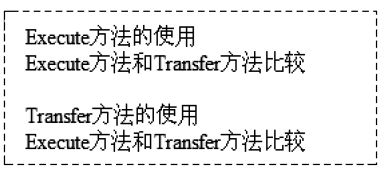


图 6-25　Execute 方法和 Transfer 方法的比较

6.5　ASP.NET 应用举例——建立网上课堂讨论区

在前面几节详细介绍了.NET 程序的结构、语法，以及服务器控件和 ASP.NET 对象，下面通过一个综合实例，将所学的知识融会贯通，完整地展示.NET 程序的应用架构。

【例 6-41】建立一个网上课堂讨论区。登录到这个讨论区的用户可以在这里畅所欲言，查看别人的发言。该实例的用户登录界面如图 6-26 所示。

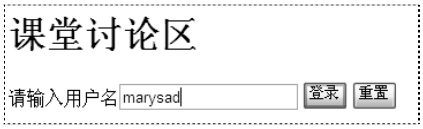


图 6-26　用户登录页面

图 6-27　输入错误提示信息

输入用户名时，要求用户名不得少于 4 个字符，否则出现如图 6-27 所示的错误提示信息。单击“登录”按钮进入讨论区的主页面，如图 6-28 所示。

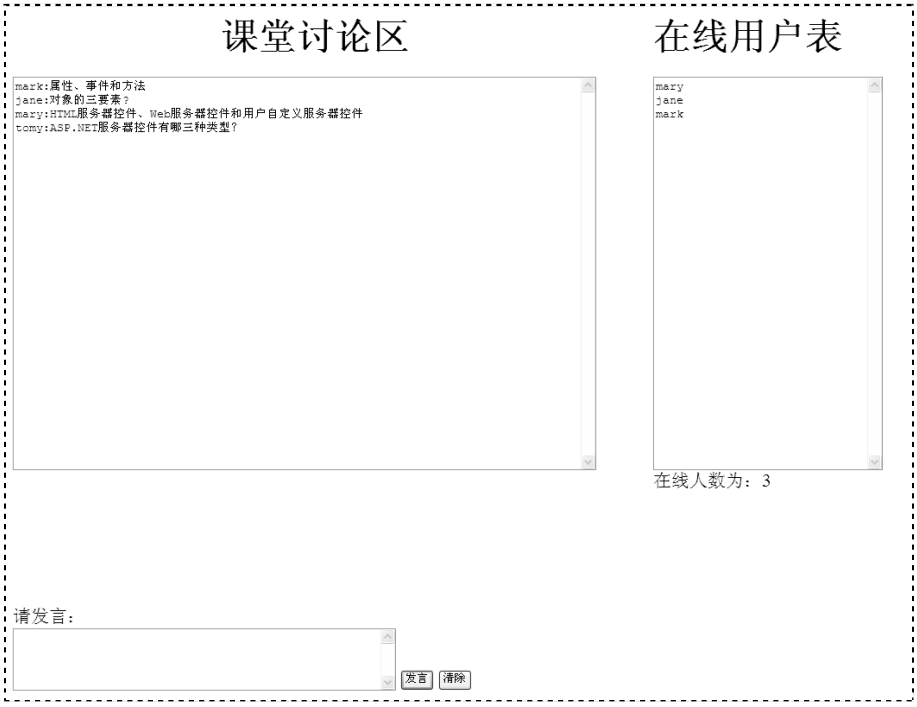


图 6-28　讨论区主页面

该应用程序由 6 个文件组成。

login.aspx	用户登录页面。
main.aspx	讨论区框架页面文件。
display.aspx	显示讨论的内容。
speak.aspx	发言信息输入页面。
show.aspx	在线用户列表。
Global.asax	定义 Session_Start()、Session_End()和 Application_Start()事件过程。

1．login.aspx 源程序

```
<%@ Page Language="C#" %>
```

```
<script runat="server">
protected void Button1_Click(object sender, EventArgs e)
{
        if (usename.Text !="" && usename.Text.Length>= 4 )
        {
            //通过登录验证，进入讨论区
            Session["username"] = usename.Text;
            Application.Lock();
            Application["username"] = Application["username"].ToString() + Session
                ["username"] + "\n";
            Application.UnLock();
            Session["flag"] = true;
            Response.Redirect("main.aspx");
        }
        else   //用户名少于 4 个字符
        {
            messsage.Visible = true;
            messsage.Text = "无效的用户名！！用户名不得少于 4 个字符!";
            usename.Text = "";
            usename.Focus();
        }
 }
protected void Button2_Click(object sender, EventArgs e)
{
        usename.Text = "";
        usename.Focus();
 }
</script>
<html>
<head runat="server">
    <title>无标题页</title>
</head>
<body>
    <h1>课堂讨论区</h1>
    <form id="form1" runat="server">
    <div>
        请输入用户名<asp:TextBox ID="usename" runat="server"></asp:TextBox>
        <asp:Button ID="Button1" runat="server" Text="登录" onclick="Button1_Click" />
        <asp:Button ID="Button2" runat="server" Text="重置" onclick="Button2_Click" /> <br />
        <asp:Label ID="messsage" runat="server" visible="false" Font-Bold="True"
          ForeColor="#FF3300"></asp:Label>
    </div>
    </form>
</body>
</html>
```

2．main.aspx 源程序

```
<%@ Page Language="C#" %>
<script runat="server">
protected void Page_Load(object sender, EventArgs e)
    {
        if ((bool)Session["flag"]!=true)
            Response.Redirect("login.aspx");
 }
```

```
</script>
<html>
  <frameset rows="680,*"    frameborder=0>
  <frameset cols="740,*" frameborder=0>
                <frame name="display" src="display.aspx" scrolling="auto" />
             <frame name="show" src="show.aspx" scrolling="auto" />
</frameset>
<frame name="speak" src="speak.aspx" scrolling="auto"    >
<noframes>
</html>
```

3．display.aspx 源程序

```
<%@ Page Language="C#" %>
<script runat="server">
protected void Page_Load(object sender, EventArgs e)
{
                  //在多行文本框中显示讨论内容
                  content.Text=Application["message"].ToString();
}
</script>
<html>
<head runat="server">
<META HTTP-EQUIV="Refresh" content="1;display.aspx">
</head>
<body>
  <form id="form1" runat="server">
  <div>
        <center><h1>课堂讨论区</h1></center>
        <asp:TextBox ID="content" runat="server" Height="455px" ReadOnly="True"
              TextMode="MultiLine" Width="676px"></asp:TextBox><br/>
</div>
</form>
</body>
</html>
```

4．speak.aspx 源程序

```
<%@ Page Language="C#" %>
<script runat="server" >
  protected void Button1_Click(object sender, EventArgs e)
{
          String usename;
          if (speakmess.Text !="" )
          {
                usename = Session["username"].ToString(); //读取用户名
                Application.Lock();
                 //将用户名、发言内容写入 Application 变量
                Application["message"] = usename + ":" + speakmess.Text
+ "\n" +Application["message"].ToString();
                Application.UnLock();
                speakmess.Text = "";
          }
  }
```

```
protected void Button2_Click(object sender, EventArgs e)
{
    speakmess.Text = "";
}
</script>
<html>
<head runat="server">
    <title>无标题页</title>
</head>
<body>
   <form id="form1" runat="server">
     <div>
     请发言：<br/>
     <asp:TextBox ID="speakmess" runat="server" Width="444px" Height="72px"
              TextMode="MultiLine" ></asp:TextBox>
     <asp:Button ID="Button1" runat="server" Text="发言" onclick="Button1_Click" />
     <asp:Button ID="Button2" runat="server" Text="清除" onclick="Button2_Click" /> <br /> <br />
     </div>
</form>
</body>
</html>
```

5．show.aspx 源程序

```
<%@ Page Language="C#" %>
<script runat="server">
protected void Page_Load(object sender, EventArgs e)
{
            //在多行文本框中显示在线用户
            content.Text=Application["username"].ToString();
            //在文本框中显示在线人数
            count.Text = "在线人数为：" + Application["usercount"].ToString();
 }
</script>
<html>
<head runat="server">
   <META HTTP-EQUIV="Refresh" content="1;show.aspx">
</head>
<body>
    <form id="form1" runat="server">
    <div>
     <left><h1>在线用户表</h1></left>
    <asp:TextBox ID="content" runat="server" Height="455px"
          TextMode="MultiLine" Width="266px" ReadOnly="True" ></asp:TextBox><br/>
    <asp:Label ID="count" runat="server" ></asp:Label>
    </div>
    </form>
</body>
</html>
```

6．Global.asax 源程序

```
protected void Application_Start(object sender, EventArgs e)
{
```

```
                Application["message"] = "";//讨论信息
                Application["usercount"] = 0;//在线用户数统计
                Application["username"] = "";//在线用户列表
    }
    protected void Session_Start(object sender, EventArgs e)
    {
            Session["username"]= "";
            Session.Timeout = 2;
            Application.Lock();
            Application["usercount"] = (int)Application["usercount"]+ 1;//在线人数加 1
            Application.UnLock();
     }
    protected void Session_End(object sender, EventArgs e)
    {
                string str1,str2;
                str1 = Session["username"].ToString();
                Application.Lock();
                Application["usercount"] = (int)Application["usercount"] - 1;//在线人数减 1
                str2 = Application["username"].ToString();
                Application["username"] = str2.Replace(str1+"\n","");
    //将退出用户名从在线用户表中去除
                Application.UnLock();
                Session.Abandon();//释放 Session 对象占用的资源
      }
```

本 章 小 结

本章介绍了 ASP.NET 程序的基本结构和语法。与 ASP 的单文件页模式相比，ASP.NET 引入了一个新的的页面模式——代码隐藏页模式，这一模式对于代码的重用、程序的调试和维护均有着重要的意义。除此之外，还可以有效地保护代码，提高程序的安全性。在 ASP.NET 应用中，使用了多种文件类型，分别起着不同的作用，其中使用最频繁的文件类型是.aspx、.aspx.vb 和.aspx.cs。命名空间是.NET 框架中又一个重要的概念，它采用树形结构管理方式，将一些提供相似功能或具有相似状态的类聚合在一起组成一个在逻辑上相关的单元，以便使用。

ASP.NET 支持的编程语言有 VB.NET、C#、Jscript.NET 等。本章重点介绍了 C#语言。从数据类型、运算符、各种控制语句到面向对象的编程等语法知识都一一进行了介绍。

从 HTML 标记发展到现在的服务器控件，它彻底地改变了页面设计的方法和数据提交的方式，为实现页面元素和编程逻辑的分离提供了便利。服务器控件不仅使用方便、功能强大，而且是服务器能够访问和操作的控件。ASP.NET 服务器控件主要分为三种类型：HTML 服务器控件、Web 服务器控件和用户自定义服务器控件。其中 Web 服务器控件又分为标准服务器控件、验证控件、导航控件、数据控件、登录控件等，这些是构成 Web 页面的重要元素。本章重点介绍了标准服务器控件。

ASP.NET 的对象是程序设计不可缺少的重要元素，这些对象由.NET Framework 中封装好的类来实现。其中包括 Page、Response、Request、Application、Session 和 Server 对象等。它们为开发者提供基本的请求、响应、会话等处理功能。

习　题　6

6.1　所有服务器控件都有哪两个属性？

6.2　与 Web 页面相关联的文件是哪两个？各有什么功能？

6.3　Web 页面可以使用哪三种控件？

6.4　命名空间是什么？Page 类位于哪个命名空间？

6.5　C#语言中，值类型和引用类型有何不同？

6.6　C#不同类型的数据进行转换的规则是什么？

6.7　声明类中的成员时，使用不同的访问修饰符表示对类成员的访问权限不同，常用的访问控制有哪些？

6.8　简述 Page 对象的 IsPostBack 和 IsValid 属性的含义，分别说明 Page 对象的 Init()事件、Load()事件和 Unload()事件何时发生。

6.9　Application 对象和 Session 对象的事件代码写在哪里？

6.10　在 ASP.NET 程序中如何获得浏览器的信息？

上机实验 6

实验 6.1　找出某区间内的所有素数。

【目的】学会 C#编程方法。

【内容】编写 C#的控制台应用程序，找出 3～100 之间的所有素数，并显示在窗口中。

【步骤】

（1）用.NET 开发软件建立一个空白的 cs 程序文件，在 Main()方法中添加求素数的程序代码。

（2）执行 cs 程序文件，查看运行结果。

实验 6.2　找出 10 个数中的最大值、最小值和平均值，并输出高于平均值的数据个数。

【目的】掌握 C#的数组定义和使用方法。

【内容】从键盘输入 10 个数据，存放在一维数组中。遍历数组中的元素找出其中的最大值和最小值，求平均值。要求编写一个 C#的控制台应用程序实现它。

【步骤】

（1）用.NET 开发软件建立一个空白的 cs 程序文件，在 Main()方法中添加程序代码。

（2）执行 cs 程序文件，查看运行结果。

实验 6.3　设计一个实现加、减、乘、除运算的控制台程序。

【目的】掌握类的定义和使用方法。

【内容】用户通过键盘输入两个数据，自动求出两个数据的和、差、积、商。要求定义一个包括 plus，minus，multiply，divide 四种方法的 Math 类。编写 C#的控制台应用程序，实现该功能。

【步骤】

（1）用.NET 开发软件建立一个空白的 cs 程序文件。首先定义一个 Math 类，然后在 Main()方法中创建该类的对象，通过对象调用类的方法。

（2）执行 cs 程序文件，查看运行结果。

实验 6.4　设计一个学生信息调查表。

【目的】学会使用标准服务器控件。

【内容】设计一个学生信息调查表页面。内容包含学号、姓名、性别、出生年月日、出生地、家庭住址等。当数据填写完成后，单击“ 确定 ”按钮，将输入的信息在页面上显示出来。单击“ 重

置”按钮，实现输入重置。程序运行界面如图 6-29 所示。

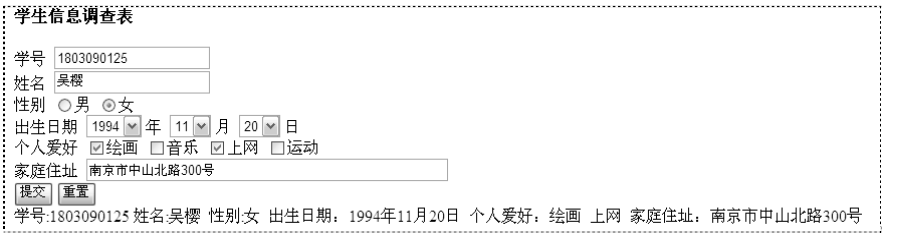


图 6-29　学生信息调查表页面

【步骤】

（1）建立一个空白的 aspx 程序文件，按图 6-29 设计学生信息调查表页面。

（2）添加“提交”命令按钮的事件过程代码，读取服务器控件中的数据。

（3）运行程序，查看结果。

实验 6.5　设计一个网页访问计数器。

【目的】学会使用 Application 对象。

【内容】编写程序使得当第 10 个用户访问该页面时，显示一条祝贺信息。

【步骤】

（1）建立一个空白的 aspx 程序文件。

（2）添加 Page_load()事件过程，设置一个 Application 变量统计访问人数。当人数为 10 时，显示一条祝贺信息。

第 7 章　Web 数据库程序设计

数据库应用系统在现有计算机应用软件中占有很大的比例。Internet 技术的飞速发展使得基于 Web 的应用需求大量涌现，越来越多的 Web 站点建立在数据库基础上，而 Web 页面和数据库的交互是众多 Web 应用程序实现的关键问题之一。本章将着重介绍基于.NET 框架的 ADO（ActiveX Data Objects）.NET 技术，讨论 ADO.NET 对数据库的访问操作。

ADO.NET 是为.NET 框架而创建的，是应用程序和数据源之间访问的桥梁，用来开发数据库应用程序。应用程序可以使用 ADO.NET 连接到 Microsoft SQL Server、Oracle 等数据源，并检索、操作和更新数据。简单地说，ADO.NET 提供了一个统一的编程模式和一组公用的类来进行任何类型的数据访问。

7.1　Web 数据库访问技术

Internet 环境下，一个 Web 应用系统一般采用 Browser/Web Server/Application Server 模式实现，Web 数据库应用系统也不例外。因此，对于应用系统来说，Web 程序设计语言、应用服务器平台及二者之间的接口技术是必不可少的。就数据库应用系统来说，应用服务器就是数据库服务器，因此 Web 访问数据库的关键在于与数据库服务器间的接口，Microsoft 公司推出的 ADO.NET 对象则是实现数据库访问的这种接口技术。ADO.NET极大地简化了 Web 数据库应用的开发工作。

Web 数据库访问一般基于 ODBC（Open Database Connectivity）或 JDBC（Java Database Connectivity）平台。ODBC 是一个数据库编程接口，由 Microsoft 公司建议并开发。它允许程序使用结构化查询语言（SQL）作为数据访问标准，应用程序可通过调用 ODBC 的接口函数来访问来自不同数据库管理系统的数据。对于应用程序来讲，ODBC 屏蔽了异种数据库之间的差异。Web 也是一类应用，和其他应用程序一样可以通过 ODBC 实现对数据库的访问。图 7-1 为采用 ODBC 访问数据库的 Web 应用系统模型。

图 7-1　ODBC 应用系统模型

JDBC 与 ODBC 一样是支持基本 SQL 功能的一个通用低层的应用程序编程接口（API），它在不同的数据库功能模块层次上提供一个统一的用户界面，只不过由于 ODBC 提供的是 C 语言 API()，而 JDBC 提供一个 Java 语言的 API()，这使得独立于 DBMS 的 Java 应用程序的开发成为可能，同时也提供了多样化的数据库连接方式。

JDBC 有两种接口：面向程序开发人员的 JDBC API 和面向低层的 JDBC Driver API。JDBC API 是一系列抽象的接口，它使应用程序员能够进行数据库连接，执行 SQL 命令，并且得到返回结果。JDBC Driver API 是面向驱动程序开发商的编程接口。

JDBC 是较早的 Web 开发平台，在 Web 应用中，嵌于网页（HTML 语言）中的 Java applets 常通过 JDBC 来访问数据库。图 7-2 为用 JDBC 实现 Web 数据库访问的方法。

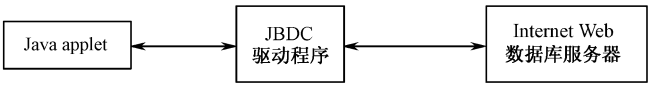


图 7-2　JDBC 技术的 Web 数据库模型

7.2　ODBC 接口

7.2.1　ODBC 接口概述

ODBC 是 Microsoft 公司建议并开发的数据库 API 标准，它为异种数据库提供统一的访问接口，使应用程序能够用结构化查询语言 SQL 访问数据库，从而为操作数据库的应用程序屏蔽了来自不同数据库管理系统的访问差异，也使数据库系统的开发不再局限于某个 DBMS。

在 Windows 平台下，ODBC 用动态链接库（Dynamic Link Libraries，DLL）调用 ODBC 驱动程序来完成对数据库的访问。对应于某一种 DBMS 有相应的 ODBC 驱动程序，当 DBMS 改变时，只要更换 ODBC 驱动程序而无须更改应用程序。

作为一种 API 标准，ODBC 主要定义了以下一些内容。

（1）ODBC 函数库。它为应用程序提供连接 DBMS、执行 SQL 语句、提取访问结果的程序接口。

（2）SQL 语法。它遵循 X/Open and SQL Access Group Call Level Interface Specification 标准。

（3）错误代码。

（4）连接、登录 DBMS。

（5）数据类型。

ODBC 接口具有相当强的灵活性，构成 SQL 语句的字符串可以在源程序中给出，或者在运行时动态生成，同一个应用程序可以存取不同的 DBMS。应用程序不必关心 ODBC 与 DBMS 底层的通信协议。

7.2.2　ODBC 的应用

一个 ODBC 应用的建立应涵盖以下内容。

（1）建立需要操作数据库的应用程序。该程序通过调用 ODBC 函数提交 SQL 语句。

（2）提供运行环境。该环境应包含数据库驱动程序，它负责处理 ODBC 函数的调用，向数据源提交 SQL 请求，向应用程序返回结果。必要时，将 SQL 语法翻译成符合 DBMS 语法规定的格式。

（3）具有由用户数据库、DBMS 等构成的可供应用程序访问的数据源。

其中，数据库驱动程序应由驱动程序管理器加载到操作系统中，在 Windows 环境下，驱动程序是一个带有入口函数库的 DLL。

对一个应用程序来说，驱动程序管理器和驱动程序的操作是不可见的，因此通过 ODBC 访问数据库的基本步骤如下：

① 创建数据源；

② 建立一个与数据源的对话连接；

③ 向数据源发出 SQL 请求；

④ 定义一个缓冲区和数据格式用于存储访问结果；

⑤ 提取结果；

⑥ 处理各种错误；

⑦ 向用户报告结果；

⑧ 关闭与数据源的连接。

7.2.3 创建并配置数据源

由于目前很多 Web 应用基于 Windows 平台，所以本节将以 Windows 环境下的 ODBC 接口配置为例，介绍如何在 Windows 环境下创建可供应用程序使用的数据源。本章后面所列举的编程示例均以此为平台。

正常安装了 Windows 操作系统之后，系统中会含有 ODBC 接口管理程序。用户可通过 ODBC 数据源管理程序或系统函数调用两种方式创建或配置数据源。下面是采用 ODBC 管理程序来创建和配置数据源的过程。

在 Windows 环境下，数据源 DSN 又分为用户 DSN、系统 DSN 和文件 DSN 三种。用户和系统 DSN 存储在 Windows 注册表中。系统 DSN 允许所有用户登录到特定的服务器上访问数据库。用户 DSN 使用适当的安全身份证明来限制数据库到特定用户的连接。文件 DSN 从文本文件中获取表格，提供对多用户的访问，并且通过复制 DSN 文件，可以轻易地从一个服务器转移到另一个服务器。在 Web 数据库设计中，一般应采用系统数据源配置。例如，配置一个 Access 数据源的过程如下。

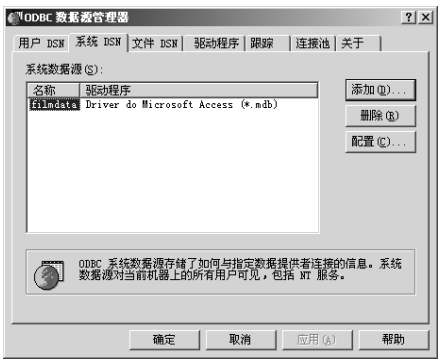


图 7-3 数据源管理器对话框

（1）在 Windows 下进入控制面板，在性能和维护中选择管理工具，打开 ODBC 数据源，屏幕上将出现如图 7-3 所示的 ODBC 数据源管理器对话框。

（2）选择“ 系统 DSN ”选项卡，即选择系统数据源。要使系统的所有应用程序都可以使用它，必须选用“ 系统 DSN ”，这是 Web 站点的需要。单击“ 添加 ”按钮，进入“ 创建新数据源 ”对话框，如图 7-4 所示。

（3）选择所使用的数据库驱动程序，如选择 Microsoft Access Driver (*.mdb)，单击“ 完成 ”按钮，出现如图 7-5 所示的对话框。

图 7-4 “ 创建新数据源 ”对话框

图 7-5 “ ODBC Microsoft Access 安装 ”对话框

（4）为该数据源命名并输入数据源名，单击“ 选择 ”按钮，为该数据源指定一个已创建好的 Access 数据库（*.mdb），该数据库应该是 Web 应用程序中即将要访问的数据库，然后回到图 7-4“ 创建新数据源 ”对话框。单击“ 确定 ”按钮，直到关闭 ODBC 管理器，此时就完成了一个数据源的创建过程。

若要修改该数据源配置，可在图 7-3 的“ 数据源管理器 ”对话框中选定要更改的数据源，再

单击“配置”按钮。

数据源创建后，即可在应用程序中使用。在 7.4 节以后的篇幅中，将介绍 ADO.NET 技术，通过已建立的数据源实现 Web 数据库应用系统。

7.3　数据库语言 SQL

7.3.1　SQL 概述

SQL（Structured Query Language）是一个被广泛采用、适用于关系数据库访问的数据库语言工业标准。它包括数据定义、数据操纵、数据查询和数据控制等语句标准。目前所有主流的 DBMS 软件商均提供对 SQL 的支持。SQL 最早出现在 1981 年，由 IBM 公司推出其雏形。1986 年由美国国家标准协会（ANSI）公布了 SQL 的第一个标准 X3.135—1986。不久，国际标准化组织 ISO 也通过了这个标准，即通常所说的 SQL—86。1989 年，ANSI 和 ISO 公布了经过增补和修改的 SQL—89。此后，又于 1992 年公布了 SQL—92，又称 SQL—2。SQL—2 对语言表达式进行了较大的扩充。

SQL 不是一种 DBMS，也不是真正、独立的计算机语言，它没有 IF 语句，也没有其他如 FOR、GOTO 这样的流程控制语句，它是一种数据库子语言，是一种控制与 DBMS 交互的语言，包含了 40 余条针对数据库管理任务的语句，这些语句可嵌入在 C 语言、Java 语言及各类脚本语言中，如 Web 系统常用的 JavaScript、VBScript、C#、VB 等语言，以扩展这些语言的数据库访问能力。SQL 具有以下 4 项功能。

（1）数据定义：定义数据模式。

（2）数据查询：从数据库中检索数据。

（3）数据操纵：对数据库数据进行增加、删除、修改等操作。

（4）数据控制：控制对数据库用户的访问权限。

SQL 语言主体大约由 40 条语句组成，每一条语句都对 DBMS 产生特定的动作，如创建新表、检索数据、更新数据等。SQL 语句通常由一个描述要产生的动作的谓词（Verb）关键字开始，如 CREATE、SELECT、UPDATE 等。紧随语句的是一个或多个子句（Clause），子句进一步指明语句对数据的作用条件、范围、方式等。

7.3.2　主要 SQL 语句

1．查询语句 SELECT

SELECT 是 SQL 的核心语句，它功能强大，和各类 SQL 子句结合，可完成多种复杂的查询操作。其语法格式如下：

```
SELECT [ALL | DISTINCT] fields_list
[INTO] new_tablename
FROM table_names
[WHERE …]
[GROUP BY…]
[HAVING…]
[ORDER BY…]
```

其中：

ALL　　//选择符合条件的所有记录，为语句默认值；

INTO　　//将查询结果放入指定新表中；
DISTINCT　　//略去选择字段中包含重复数据的记录；
Fields_list　　//用“，”隔开的字段名列表，列出需查询的记录字段，可用“*”代替所有字段；
FROM　　//SQL 子句，指定 SQL 语句所涉及的表（Table）；
Table_names　　//SELECT 语句所涉及的表名，用“，”隔开；
WHERE　　//SQL 子句，指定查询结果应满足的条件；
GROUP BY　　//SQL 子句，按照指定的字段将查询结果分组；
HAVING　　//SQL 子句，指定查询结果分组后需满足的条件，只有满足 HAVING 条件的分组才会出现在查询结果中；
ORDER BY　　//SQL 子句，指定查询结果按哪些字段排序，是升序（ASC）还是降序（DESC）。

【例 7-1】 用最基本的 SELECT 语句实现从 department 表中检索出所有部门的相关资料。

```
SELECT dept_no,dept_name,location
FROM department
```

该语句执行后的结果为：

dept_no	dept_name	location
001	research	Dallas
002	accounting	Seattle
003	marketing	Dallas

【例 7-2】用 WHERE 子句指定查询条件，要求列出位于 Dallas 的所有部门：

```
SELECT dept_no,dept_name
FROM department
WHERE location= 'Dallas'
```

语句执行后的结果如下：

dept_no	dept_name
001	search
003	marketing

【例 7-3】用 INTO 子句从雇员表 employee 中将属于 001 部门的雇员信息检索出来，并放入一个新数据库表 researchemp 中，新表按 emp_name 字段升序排列。

```
SELECT *
INTO researchemp
FROM employee
WHERE dept_no= '001'
ORDER BY emp_name ASC
```

2．插入数据语句 INSERT

INSERT 可添加一个或多个记录至一个表中。INSERT 有两种语法形式：

（1）INSERT INTO target [IN externaldatabase] (fields_list)
{DEFAULT VALUES|VALUES（DEFAULT|expression_list）}

（2）INSERT INTO target [IN externaldatabase] fields_list
{ SELECT…|EXECUTE…}

其中，① target 是追加记录的表（Table）或视图（View）的名称；② externaldatabase 是外部数据库的路径和名称；③ expression_list 是要插入的字段值表达式列表，其个数应与记录的字段个数一致，若指定要插入值的字段 fields_list，则应与 fields_list 的字段个数一致。

使用第（1）种形式将一个记录或记录的部分字段插入到表或视图中。第（2）种形式的 INSERT 语句插入来自 SELECT 语句或来自使用 EXECUTE 语句执行的存储过程的结果集。

【例 7-4】用第（1）种 INSERT 形式，将一个雇员的信息插入到雇员表中。

```
INSERT INTO employee VALUES('255001', 'Ann', 'Jones', 'd3')
```

【例 7-5】用第（2）种 INSERT 形式，将从部门表 department 中检索出的位于 Dallas 的部门记录插入到 dallas_dept 表中。

```
INSERT INTO dallas_dept(dept_no,dept_name)
SELECT dept_no,dept_name
FROM department
WHERE location='Dallas'
```

3．删除数据语句 DELETE

DELETE 从一个或多个表中删除记录。语法格式如下：

```
DELETE
FROM table_names
[WHERE…]
```

【例 7-6】删除表 works_on 中所有经理的记录。

```
DELETE FROM works_on
WHERE job='manager'
```

【例 7-7】从数据库表中删除雇员 Ann 的相关记录。

```
DELETE FROM works_on
WHERE emp_no IN
(   SELECT emp_no
      FROM    employee
      WHERE emp_no='Ann')
```

4．更新数据语句 UPDATE

UPDATE 语句用来更新表中的记录。语法格式如下：

```
UPDATE table_name
SET Field_1=expression_1[,Field_2=expression_2…]
[FROM table1_name|view1_name[,table2_name|view2_name…]]
[WHERE…]
```

其中，① Field 是需要更新的字段；② Expression 是要更新字段的新值表达式。

【例 7-8】更新 employee 表中雇员 Jhon 的部门。

```
UPDATE employee
SET dept_no='d3'
WHERE emp_fname=' Jhon'
```

【例 7-9】将雇员 jones 的岗位置空。

```
UPDATE works_on
SET job=NULL
FROM works_on,employee
WHERE emp_lname='jones'
AND works_on.emp_no = empoyee.emp_no
```

7.4　ADO.NET 数据库组件

ADO.NET 是由 ASP 平台下的 ADO 技术发展起来的，以配合 ASP.NET 平台下的数据库应用

开发，用于帮助.NET 框架下的应用程序从 Microsoft SQL Server、Oracle、Access 等不同的数据源中快速访问数据。

ADO.NET 由一组.NET 框架中的类库构成，这是一组数据源连接、提交查询和处理结果的类的集合。ADO.NET 也提供了很多数据访问、数据操作、数据显示的控件，通过.NET 框架数据提供程序 Provider 所提供的应用程序编程接口（API），可以轻松地访问各种数据源，包括 OLEDB 和 ODBC 支持的数据，同时大大简化对数据库的操作。ADO.NET 提供的许多功能可替代开发过程中大量的编码，从而提高开发效率。

ADO.NET 建立在 XML 的基础上，在非 Windows 平台下，以往的 ADO 记录集不能直接使用，从而使协同操作能力受到限制。在 ADO.NET 中，可将数据转换成 XML 格式，以便于网络传输和异构平台的操作。因此，通过 ADO.NET 不仅能访问关系型数据库中的数据，也能访问 XML 格式数据。

7.4.1 ADO.NET 组件模型

ADO.NET 将数据操作与数据访问分开，它包含两个核心组件：数据集 DataSet 和.NET 数据提供者 Provider。DataSet 是非连接的、位于内存中的数据存储对象，主要负责对数据的操作；而 Provider 是一套特有的组件，用于访问各种类型的数据源，它主要负责对数据源的读写和操纵。ADO.NET 组件模型如图 7-6 所示。

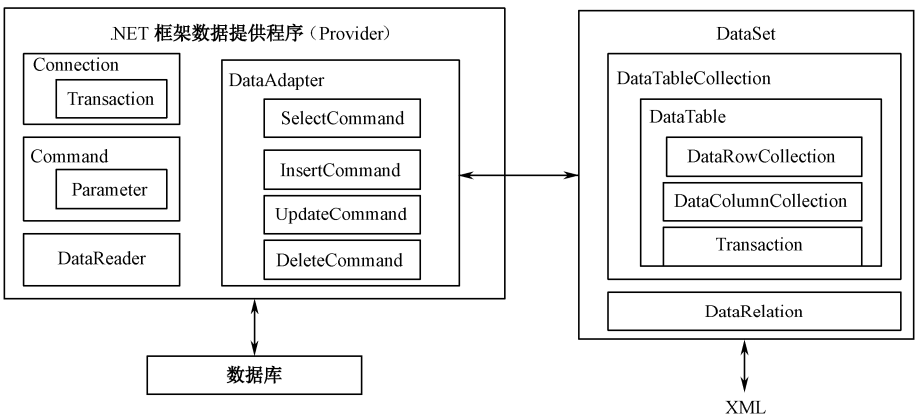


图 7-6　ADO.NET 组件模型

可以看出 ADO.NET 组件模型由两大部分组成：.NET 框架数据提供程序 Provider 和数据集 DataSet。

1．数据提供程序 Provider

.NET 数据提供程序 Provider 提供了一组连接、执行数据库访问命令的对象集，主要包含 4 个核心对象：Connection、DataReader、Command、DataAdapter。其中，Connection 对象提供与数据源的连接；Command 对象对数据源执行数据库访问命令，用于返回、修改数据，执行存储过程；DataReader 对象用于从数据源中读取数据；DataAdapter 提供连接 DataSet 对象和数据源的桥梁，它使用 Command 对象在数据源中执行 SQL 命令，以便将数据加载到 DataSet 中，并使 DataSet 中数据的更改与数据源保存一致。这组对象中实现连接状态下的数据库访问。

除此之外，.NET 数据提供程序还包含以下对象：

① Transaction 对象：用于数据访问的事务处理，可在其生存周期内提交或取消对数据所做的更改；

② CommandBuilder 对象：自动生成 DataAdapter 的命令属性或从存储过程中派生参数信息，并填充 Command 对象的 Parameters 集合，用于指定 SQL 参数；

③ ConnectionStringBuilder 对象：提供一种用于创建和管理由 Connection 对象使用的连接字符串内容的简单方法；

④ Parameter 对象：定义命令和存储过程的输入、输出和返回值参数；

⑤ Exception 对象：异常处理对象，在数据源中遇到错误时返回。对于在客户端遇到的错误，.NET Framework 数据提供程序引发一个.NET Framework 异常；

⑥ Error 对象：发布数据源返回的警告或错误中的信息；

⑦ ClientPermission 对象：为.NET Framework 数据提供程序代码访问安全属性。

数据提供程序分为不同的类型以适应不同的数据源，如表 7-1 所示。

表 7-1　.NET Framework 数据提供程序的类型

.NET Framework 数据提供程序类型	说　明
.NET Framework SQL Server 的数据提供程序	提供对 Microsoft SQL Server 7.0 或更高版本中数据的访问。使用 System.Data.SqlClient 名称空间
.NET Framework OLE DB 的数据提供程序	提供对使用 OLE DB 公开的数据源中数据的访问。使用 System.Data.OleDb 名称空间
.NET Framework ODBC 的数据提供程序	提供对使用 ODBC 公开的数据源中数据的访问。使用 System.Data.Odbc 名称空间
.NET Framework Oracle 的数据提供程序	适用于 Oracle 数据源。用于 Oracle 的.NET Framework 数据提供程序支持 Oracle 客户端软件 8.1.7 和更高版本，并使用 System.Data.OracleClient 名称空间
EntityClient 的数据提供程序	提供对实体数据模型(EDM)应用程序的数据访问。使用 System.Data.EntityClient 名称空间

2．数据集 DataSet

数据集 DataSet 的一组对象则用于在非数据库连接状态下，存储访问独立于任何数据源的数据，可以是 Provider 对象提供的数据，也可以是来自于 XML 的数据。

DataSet 包含一个或多个 DataTable 对象的集合，这些 DataTable 对象由数据行、数据列和主键、外键、约束及有关 DataTable 对象中数据的关系信息组成。

DataTable 对象对应一个逻辑表，它有两个重要的属性 Rows 和 Columns，对应表中的行和列，定义一系列描述该列结构的属性。Constraint 对象提供本地存储数据的约束方式。DataRelation 对象定义 DataSet 中不同 DataTable 对象之间的关系，并可通过 DataRow 对象获取。DataView 对象可用不同方式查看 DataTable 对象中的数据。

7.4.2　ADO.NET 的数据访问模式

ADO.NET 提供两种数据访问模式：一种为连接模式（Connected），另一种为非连接模式（Disconnected），如图 7-7 所示。

连接模式的对象模型直接与数据库通信，可采用数据库访问命令，检索、更新、插入、删除数据库中数据，如采用 SQL 的 Select 命令检索数据库表。

非连接模式对象模型允许用户脱机处理，数据操作发生在 DataSet 对象集的 DataTable 中，DataSet 位于内存中，而非数据库中。

1．连接模式下的数据库访问

应用程序使用 ADO.NET 访问数据库时，首先需要用 Connection 对象连接数据库，然后用

Command 对象对数据库进行操作，Command 对象的执行结果可以被 DataReader 对象读取，也可以被 DataAdapter 对象用来填充 DataSet 对象。当 DataReader 读取时，每次只读一条数据记录，而 DataAdapter 对象则是读取命令执行结果的所有数据，并可将其填充给 DataSet。因此 DataAdapter 对象是 DataSet 对象与数据库的桥梁。这个过程如图 7-8 所示。

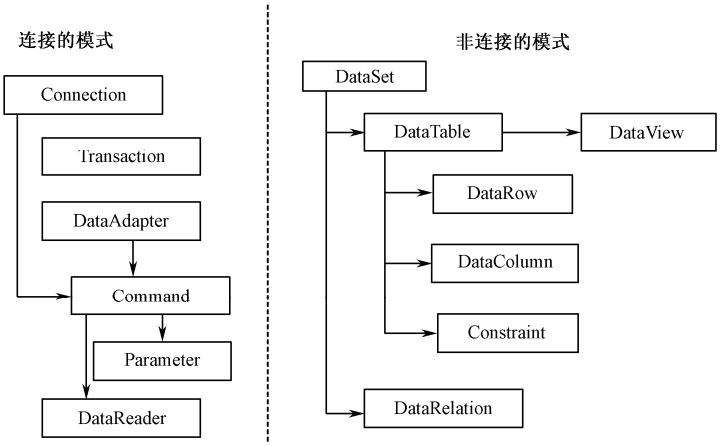


图 7-7　两种数据访问模式

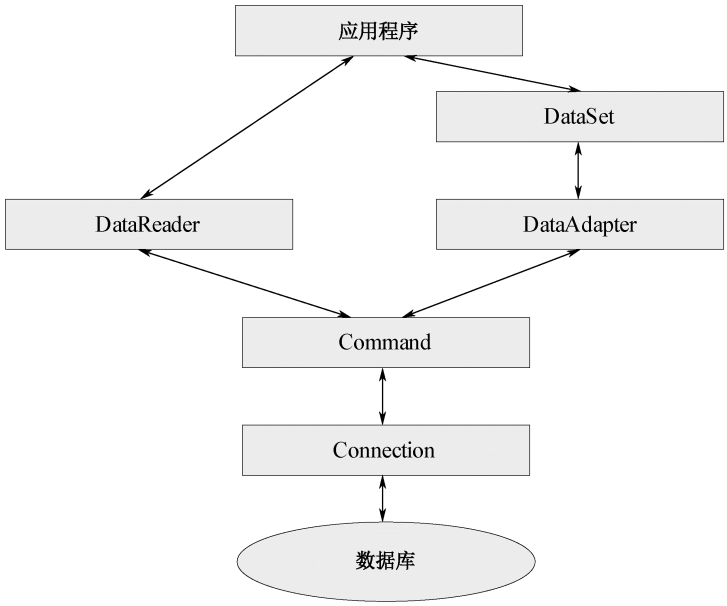


图 7-8　ADO.NET 连接数据库的过程

连接模式下访问数据库的基本步骤如下。

① 定义相应的命名空间。命名空间（NameSpace）是多个对象的逻辑分组，命名空间记录了对象的名称与所在的路径，以便编译器加载这些对象。常用的命名空间包括：

- System.Data：是 ADO.NET 的核心，包含所有数据提供程序使用的类，如 DataSet、DataTable、DataRow 等，在编写 ADO.NET 程序时，必须定义该命名空间；
- System.Data.SQLClient：当使用 Microsoft SQL Server.NET 数据提供程序连接 SQL Server 7.0 以上版本数据库时须定义使用该命名空间；
- System.Data.Odbc：当使用 Microsoft ODBC.NET 数据提供程序连接 ODBC 数据源连接的

数据库时使用；

- System.Data.OracleClient：当使用 Oracle.NET 数据提供程序连接 Oracle 数据库时使用。

② 建立与数据库的连接。可以使用 SqlConnection 或 OleDbConnection 对象等。SqlConnection 对象管理与 SQL Server 7.0 版或更高版本的连接；OleDbConnection 对象实现基于 OLEDB 访问的任何数据源的连接。SqlConnection 对象特定于 SQL Server，由于不必通过 OLEDB 层，所以比 OleDbConnection 效率高。SqlConnection 与 SqlCommand 和 SqlAdapter 对象一起使用。

③ 使用 Command 对象执行 SQL 命令，对数据库实现数据检索、增删改等。

④ 若执行的是检索命令，则使用 DataReader 对象对检索结果进行读取。

⑤ 关闭与数据库的连接。

2．非连接模式下的数据库访问

非连接模式下，数据的操作可以不需要与数据库服务器一直保持连接，只有当客户机需要将更新的数据传回到服务器时再重新连接，可以同时支持更多并发的客户机。这种模式依赖于 DataSet 对象集。 DataSet 对象提供了和关系数据库一样的关系数据模型，在非连接的方式下，可以直接操纵 DataSet 对象中的 DataTable 对象。DataTable 对象表示保存在本机内存中的表，它提供了对表中行列数据对象的各种操作。

非连接模式下应用程序进行数据访问的步骤如下。

① 定义相应的命名空间。

② 使用 Connection 对象连接数据库。

③ 使用 DataAdapter 对象执行对数据库的访问。

④ 把 DataAdapter 对象访问数据库的结果数据填充到 DataSet（数据集）对象中。

⑤ 关闭 Connection 对象。

⑥ 执行 DataSet 对象中数据需要的操作，如增删改等。

⑦ 如有必要，重新启动 Connection 对象连接数据库，使用 DataAdapter 对象将更新后的 DataSet 回写到数据库。

⑧ 关闭 Connection 对象。

7.5　ADO.NET 对象

ADO 的对象集合可以参考图 7-6 和图 7-7，因篇幅关系只介绍常用的对象。本章采用 SQL Server 作为数据源，故示例代码的对象名称前加 Sql，如 SqlConnection、SqlCommand 等。

7.5.1　Connection 对象

Connection 对象建立与所要访问的数据源的关联关系，它具有一组属性和方法，用于表示、维护这个关联关系，打开连接后，可以用 command 对象等完成对数据库的操作。

在 ADO.NET 对象模型中，所有 Connection 类都派生自 System.Data.Common 命名空间中的 DbConnection 类。本章主要以 SqlConnection 类为例，用于 Microsoft SQL Server 数据源。

1．Connection 对象的常用属性和方法

① ConnectionString 属性：设置连接数据源的一些控制信息。

② State 属性：指明 Connection 对象所处的状态，这些状态有关闭、打开、正在连接、正在执

行命令等。

③ ConnectionTimeout 属性：设置对象建立连接等操作失败时的等待时间。

④ Database 属性：已连接或将要连接的数据库名称。

⑤ ServerVersion 属性：返回数据源的版本（只读）。

⑥ Open()方法：建立到数据源的物理连接。

⑦ CreateCommand()方法：为当前连接生成 Command。

⑧ BeginTransaction()方法：开始该连接上的一个事务。

⑨ Close()方法：关闭一个连接。在对 Connection 对象操作结束时，使用它释放所有与之关联的系统资源。

⑩ ChangeDatabase()方法：改变当前连接的数据库。

2．创建、打开和关闭数据库连接

在运行时创建 Connection 对象通常有两种方式。一种方式可以使用无参数构造函数简单地生成一个未初始化的 Connection 对象，然后设置其 ConnectionString 属性，例如：

```
SqlConnection conn = new SqlConnection();
conn.ConnectionString="Data Source=(local);Initial Catalog=student; User ID=sa; Password=12345";
```

这种方式下，先创建一个无参数的 SqlConnection 连接对象 conn，然后设置该对象的连接字符串属性 ConnectiontionString，该属性定义了连接数据库必要的控制信息，这些信息包括数据源名称、数据库用户名和用户口令。

另一种方式为使用带连接字符串参数的构造函数创建 Connection 对象，例如：

```
string    strConn="Data Source=(local);Initial Catalog=student; User ID=sa; Password=12345";
SqlConnection conn = new SqlConnection(strConn);
```

这种方式先定义一个连接控制字符串，然后再创建一个带有该字符串参数的连接对象。

创建 Connection 对象后，其被初始化为“关闭”状态，必须进行打开操作，才能对数据源中的数据进行访问，而当所有对数据源的操作完成后，应关闭这个 Connection 对象。关闭连接是因为大部分数据源仅支持有限数量的打开连接，同时打开连接也会占用系统资源。

打开连接对象可采用 Connection 对象的 Open()方法，而关闭数据库则使用 Close()方法。

具体实现时，首先可以使用控制面板中的管理工具创建一个 ODBC 数据源 DSN，然后用代码创建一个到数据源的连接。例 7-10 是在已创建了一个 SQL Server 数据源的情况下，用 C#代码实现到数据库 student 的连接。

【例 7-10】数据库连接示例。

```
using System.Data;
using System.Data.SqlClient;
…
string strConn = "Data Source=(local);Initial Catalog=student; ";
strConn += "User ID=sa; Password=12345";
//定义一个连接数据库字符串，连接到 student 数据库，用户名是 sa，密码为 12345
SqlConnection conn = new SqlConnection(strConn);
//创建 SqlConnection 数据库连接对象
using(conn)
{
    conn.Open();      //打开数据库连接
    …   //这里为对数据源操作的代码
    conn.close();   //关闭数据库连接
}
```

本例中，开始的两条 using 语句定义了 ADO.NET 对象所在的命名空间，conn 为创建的一个 SqlConnection（连接）对象，以用户 sa 的身份连接了一个已建立的数据源“ student ”。对数据库的访问一般要使用上述创建和连接代码，这也是连接模式下数据访问的基本方式。打开连接后对数据库数据的访问操作将在介绍ADO.NET其他对象时加以描述。

上述代码不管在执行数据库命令时是否有错误产生，using 语句都会确保强制关闭连接，同时还将撤销（dispose）Connection 对象，如果要再次使用该 Connection 对象，那么需要对其重新进行初始化。

7.5.2　Command 对象

1．Command 对象的常用属性和方法

Web 中对数据库数据的查询、更新、插入、删除等操作通过执行 SQL 语句来实现，ADO 的 Command 对象用来表示一个 SQL 语句字符串并执行这个命令串。通过 Command 对象可执行一句或一批 SQL 查询，也可驱动 SQL Server 的存储过程。而运用 Command 对象参数集合和调用存储过程的能力，可以进行更强大的数据库访问操作。

Command 对象的常用属性和方法如下。

① Connection 属性。指定与 Command 对象关联的连接对象。

② CommandText 属性。定义一个可执行的命令串，可被 Execute()方法执行。

③ Type 属性。指定命令串的类型。类型包括：Text（缺省）、StoreProcedure 和 TableDirect。

④ CommandTimeout 属性。指定 Adapter 对象等待命令执行的时间，缺省为 30 秒。

⑤ Parameters 属性。命令的参数集合。

⑥ Transaction 属性。定义一个事务。

⑦ ExecuteNonQuery()方法。用于执行一个不返回结果的命令。

⑧ ExecuteReader()方法。执行 CommandText 定义的命令，并获取结果至 DataReader 对象。

⑨ ExecuteScalar()方法。执行命令，只获取结果中的第 1 行第 1 列数据，例如执行一条 Select 语句，只获取查询结果的第 1 个值。

2．用 Command 对象执行 SQL 语句

要采用 Command 对象完成数据库操作，首先要创建一个 Command 对象实例，然后指定其 Connection（命令关联的连接）属性、CommandText（要执行的命令串）属性，必要时可指定 CommandType（命令串的类型）属性以优化执行时的性能，然后用命令对象的 Execute()方法完成对 SQL 语句的执行。

【例 7-11】用 Command 对象的 Execute()方法从 sales 表中检索所有记录。

```
string strConn, strSQL;
strConn="Data Source=(local);Initial Catalog=student; User ID=sa; Password=12345";
SqlConnection conn = new SqlConnection(strConn);
strSQL = "SELECT * FROM courses";   //定义一个 SQL 查询命令串
SqlCommand cmd = new SqlCommand();   //建立一个命令对象
cmd.Connection = conn;     //指定命令对象的连接属性
cmd.CommandText = strSQL;   //指定命令对象可执行的命令串
conn.Open();
SqlDataReader myrdr=cmd.ExecuteReader();     //用 ExecuteReader（）方法执行数据库命令
conn.close();
```

本例中，创建了一个 Connection 对象实例 conn 和一个 Command 对象实例 cmd，然后

“ cmd.Connection = conn; ”语句指定 cmd 命令执行针对的是连接串 conn，“ cmd.CommandText = strSQL ”语句则指定了一条由 strSQL 定义的可执行的命令，“ cmd.ExecuteReader() ”语句执行命令并将结果返回给 DataReader 型对象 myrdr，即将 courses 表的所有记录返回给 myrdr 对象存储。DataReader 对象将在 7.5.3 中介绍。

除了像本例中创建一个无参命令 cmd 对象，还可以通过参数指定，直接创建一个指定了命令串、连接的命令对象，如代码：

```
SqlCommand cmd = new SqlCommand(strSQL,conn);
```

创建了一个命令对象 cmd，该对象的 CommandText 属性值为 strSQL，且连接为 conn。

另外 Command 对象的创建也可以采用 Connection 对象的 CreateCommand 属性，例如：

```
SqlConnection conn = new SqlConnection(strConn);
cmd=conn.CreateCommand();
```

同样可以创建一个 Command 对象。

如果对数据库的操作不涉及结果的返回，如 update 和 delete，可以采用 Command 命令的 ExecuteNonQuery()方法。如例 7-12 所示。

【例 7-12】用 Command 对象的 ExecuteNonQuery()方法从 sales 表中删除 productId 为 100415 的记录。

```
string strConn, strSQL;
strConn="Data Source=(local);
Initial Catalog=student; User ID=sa; Password=12345";
SqlConnection conn = new SqlConnection(strConn);
strSQL＝“DELETE ＊ FROM sales WHERE productId=‘100415’”;   //定义可执行的命令串
SqlCommand cmd = new SqlCommand(strSQL, conn);   //建立一个命令对象
conn.Open();
cmd.ExecuteNoQuery();          //执行 DELETE 命令
conn.close();
```

3．Command 对象 Parameters 属性的使用

Command 对象的 Parameters 属性允许对数据库访问的命令执行时可以使用命令参数，例如，执行查询语句时，查询条件来自用户输入和其他函数的调用参数。

Parameters 属性的数据类型为 ParameterCollection，实际为参数对象 Parameter 的集合。Parameter 对象也有自己的属性和方法。

① ParameterName 属性：参数名称，形式为以@开头的字符串，如@stuName。

② SqlDbType 属性：参数数据类型，依赖于数据提供程序，为 SqlDbType 或 OleDbType。

③ Size 属性：参数中数据的最大字节数。

④ Direction 属性：指定参数是输入还是输出参数。分别为：ParameterDirection.Input（默认值）、ParameterDirection.InputOutput、ParameterDirection.Output、ParameterDirection. ReturnValue（该参数为返回值）。

⑤ Add()方法：添加参数对象到命令对象的参数集合中。Add()方法可以有 Add(SqlParameter)，Add(String, SqlDbType)，Add(String, SqlDbType, Int32)，Add(String, SqlDbType, Int32, String)几种形式。其中 SqlParameter 是参数对象，string 为参数名称，如@stuName，int32 为参数的范围，为整型值，而 SqlDbType 可以是 SqlDbType.NVarChar，SqlDbType.Int，SqlDbType.DateTime，SqlDbType.Bit，SqlDbType.Money，SqlDbType.Text，SqlDbType.Image 等。

【例 7-13】根据用户输入的学生姓名和所在系名查询数据库中该生的信息。假设用户输入的学生姓名是“ 李明 ”，系名为“ 计算机 ”，用 SqlCommand 对象 Parameters 属性的 Add(SqlParameter)

方法实现查询。

```
string strConn, strSQL;
strConn="Data Source=(local);Initial Catalog=student; User ID=sa; Password=12345";
SqlConnection conn = new SqlConnection(strConn);
strSQL = " SELECT * FROM student WHERE name=@stuName and dept=@deptName ";
SqlCommand    cmd = new SqlCommand(strSQL, conn);
parm1 = new SqlParameter(); //新建一个参数对象
parm2= new SqlParameter();
parm1.ParameterName = "@ StuName ";
parm1.Value = "李明";
parm2.ParameterName = "@ deptName ";
parm2.Value = "计算机";
cmd.Parameters.Add(parm1);   //将第 1 个参数对象加入到命令对象的参数集合
cmd.Parameters.Add(parm2);   //将第 2 个参数对象加入到命令对象的参数集合
using ( conn)
{
      conn.Open();
      SqlDataReader myrdr = cmd.ExecuteReader();   //Command 对象执行带参的查询操作。
      ...
}
```

也可以用 Add(String, SqlDbType)实现加入命令对象参数集合，例如：

```
SqlParameter parm = command.Parameters.Add("@stuName", SqlDbType.VarChar);
parm.Size = 20;              //定义字符串长度
parm.Value = "李明";       //参数赋值
```

4．用 Command 对象调用存储过程

存储过程是 Microsoft SQL Server 的一个概念，如同其他程序设计语言中的过程、函数或子程序的概念，它由一系列的 SQL 语句组成，常用来完成一个特定的功能，便于共享及程序模块化。存储过程在 SQL Server 中通过如下语句建立，可被其他应用程序调用。

```
CREATE PROCEDURE  过程名  (输入、输出参数表)
AS
SQL 语句序列
```

在 Web 数据库程序设计中使用 SQL 存储过程有下列好处。

① SQL 存储过程的执行比 SQL 命令快得多。当一个 SQL 语句包含在存储过程中时，服务器不必每次执行它时都要分析和编译它。

② 在多个网页中可以调用同一个存储过程，使站点易于维护。

③ 一个存储过程可以包含多个 SQL 语句，这意味着可用存储过程建立复杂的查询。

④ 存储过程可以接收和返回参数，这是复杂数据库访问功能实现的必要基础。

（1）存储过程的调用形式

存储过程的调用也是一个 Command 对象的执行。Command 对象调用存储过程时，需要首先指定对象的 CommandType 属性值为 CommandType.StoredProcedure，其次指定对象的 CommandText 属性值为具体的存储过程的名称，执行该 Command 对象的 Execute()方法系列便可完成存储过程的调用。

【例 7-14】实现对存储过程 HitCount 的调用示例。

```
string strConn, strSQL;
strConn="Data Source=(local);Initial Catalog=student; User ID=sa; Password=12345";
SqlConnection conn = new SqlConnection(strConn);
SqlCommand    cmd = new SqlCommand(); //创建一个命令对象
```

```
cmd.CommandType = CommandType.StoredProcedure;  // 指定 Command 类型为调用存储过程
cmd.CommandText= "HitCount";  //指定要调用的存储过程名称
cmd.Connection=Conn;
try
{
conn.Open();
SqlDataReader rdr = cmd.ExecuteReader();  //执行调用存储过程
      …
}
finally
{
conn.Close();
}
```

本例中，HitCount 是一个存储过程名，当指定了 CommandType 是一个存储过程时，cmd.ExecuteReade()执行对存储过程 HitCount 的调用。

采用 try…finally 结构，则不管数据库命令的执行中是否产生错误，语句中的 finally 子句都会确保强制关闭数据源连接。

（2）存储过程的参数传递

一个存储过程通常要根据调用者传入的参数来完成相应的操作，很多情况下还需要返回执行结果及状态信息。Command 对象的 Parameters 参数集合同样可实现存储过程的参数传递。

【例 7-15】利用对一个存储过程 LoginCheck 的调用，验证登录用户的合法性。该存储过程在调用时，须给定两个入口参数——登录用户名 UserName 和密码 Password。LoginCheck 存储过程在校验完后将返回校验结果 Result。

```
string strConn, strSQL;
strConn="Data Source=(local);
Initial Catalog=student; User ID=sa; Password=12345";
SqlConnection conn = new SqlConnection(strConn);
SqlCommand   cmd = new SqlCommand();  //创建一个命令对象
cmd.Connection=Conn;
cmd.CommandText= "LoginCheck" ;
cmd. CommandType= CommandType.StoredProcedure; //  指定 Command 类型为调用存储过程
UserParam=new SqlParameter(“@UserName”, SqlDbType.VarChar, 30);
  //创建一个参数对象 UserParam，字符型，长度为 30
cmd.Parameters.Add(UserParam); //将参数对象 UserParam 添加到命令对象的参数集合中
PassParam=new SqlParameter(“@Password”, SqlDbType.VarChar, 30);
  //与数据库通信，以管理连接和事务。
cmd.Parameters.Add(PassParam);  //添加口令参数
ResultParam=new SqlParameter(“Result”, adVarChar, adParamOutput,30);
ResultParam.Direction = ParameterDirection.ReturnValue;  //指定该参数为返回参数
cmd.Parameters.Add(ResultParam);
UserParam.Value = “Jhon";  //为参数赋值
PassParam.Value=“Hello"
try
{
conn.Open();
SqlDataReader rdr = cmd.ExecuteReader();  //执行调用存储过程
      …
}
finally
{
```

```
        conn.Close();
        }
```

本例中，通过向命令对象的参数集合中添加两个输入型参数对象 Userparam、PassParam 和一个输出型参数对象 ResultParam 来实现与存储过程 LoginCheck 的参数传递。

7.5.3　DataReader 对象

在前面的例子中，已经使用过 DataReader 对象执行对数据库访问结果的提取，获取查询命令执行的结果集。该对象可以通过调用 Command 对象的 ExecuteReader()方法来获得，方式如下：

```
        DataReader rdr = cmd.ExecuteReader()；
```

其中，rdr 为一个 DataReader 对象，cmd 为一个命令对象。当只需要顺序地读取数据（数据库记录）而无须更新、增加、删除数据时，可以使用 DataReader 对象。

DataReader 对象对应当前读取的数据库数据（记录行），当从结果集中读取一行后，记录游标移动到下一数据行，而前一行将不再可用。如果需要再次读取前一行数据，必须另创建一个 DataReader 对象。DataReader 对象在读取数据的时候限制了每次只读取一行，且只能读取。

DataReader 对象的常用属性和方法包括：

① FieldCount 属性：DataReader 包含的字段数；

② Item 属性：当前行的列内容；

③ HasRows 属性：指示 command 对象执行后是否有结果（记录行）返回；

④ IsClosed 属性：指示 DataReader 是否已关闭；

⑤ RecordsAffected 属性：指示 command 对象执行后影响的记录数；

⑥ Read()方法：读取当前记录行数据，读完后指针移到下一个记录。结果集若有数据，返回 True，否则返回 False；

⑦ NextResult()方法：移动到下一个记录；

⑧ GetName()方法：根据记录字段的序号返回字段名称；

⑨ GetOrdinal()方法：根据字段的名称返回字段的序号；

⑩ GetString()方法：通过字段的位置来返回字段值；

⑪ GetSchemaTable()方法：获取 DataReader 的结构信息，包括字段名和数据类型，即对应的数据库表结构；

⑫ GetFieldType()方法：获取指定序号字段的数据类型；

⑬ GetSqlValue()/GetValue()方法：根据字段序号返回该字段的值；

⑭ GetValues()方法：获取当前记录行的内容。

1．获取结果集数据

当执行查询命令后，有多种方法都可以从 DataReader 对象中获取其当前所表示的数据行的字段值。可以通过字段的名称来返回该字段的值，例如：

```
        string name = (string) rdr ["name" ];
```

由于该字段的值是以 Object 类型返回。因此，在将该返回值赋值给字符串变量之前，必须对其进行显式的类型转换。

也可以通过字段的位置来返回该字段的值，例如：

```
        string name = (string) rdr [1];
```

该字段的值也是以 Object 类型返回，使用前也必须对其进行显式的类型转换。

```
        string name = rdr.GetString(1);
```

该方法得到的返回值的类型是字符串，不用对返回结果进行任何类型转换。

【例 7-16】在数据库中查询课程信息，并显示课程代码和名称。

```
string strConn, strSQL;
strConn="Data Source=(local);Initial Catalog=student; User ID=sa; Password=12345";
SqlConnection conn = new SqlConnection(strConn);
strSQL = "SELECT * FROM courses";          //定义一个 SQL 查询命令串
SqlCommand cmd = new SqlCommand();     //建立一个命令对象
cmd.Connection = conn;            //指定命令对象的连接属性
cmd.CommandText = strSQL;       //指定命令对象可执行的命令串
using ( conn)
{
conn.Open();
SqlDataReader rdr = cmd.ExecuteReader();
while (rdr.Read())     //用 Read 方法读取结果集中当前记录，有记录读取则返回为真
   {
     Console.WriteLine(String.Format("{0}, {1}",  rdr [0], rdr [1]));
     //将 DataReader 对象序号为 0 和 1 的字段对应的信息输出显示
    }
rdr.Close();         //关闭 SqlDataReader 对象
conn.close();
    }
```

当执行完检索命令后，DataReader 对象 rdr 位置 0 和 1 分别对应记录头两个字段的内容，也即课程代码和课程名称。

2．获取结果集结构

DataReader 对象表示查询的结果集，对应于数据库表的记录行，如果要获取数据记录的结构信息，则可以使用 DataReader 对象有关字段和行的属性和方法，如 FieldCount、GetName()、GetFieldType()等。

【例 7-17】在数据库中查询课程信息，并显示课程代码和名称。

```
string strConn, strSQL;
strConn="Data Source=(local);Initial Catalog=student; User ID=sa; Password=12345";
SqlConnection conn = new SqlConnection(strConn);
strSQL = "SELECT * FROM courses ";          //定义一个 SQL 查询命令串
SqlCommand cmd = new SqlCommand();     //建立一个命令对象
cmd.Connection = conn;          //指定命令对象的连接属性
cmd.CommandText = strSQL;       //指定命令对象可执行的命令串
using ( conn)
{
conn.Open();
SqlDataReader rdr = cmd.ExecuteReader(CommandBehavior.SchemaOnly);
//只返回结构

//以下根据字段数循环，对每个数据库字段
for (int intField = 0; intField < rdr.FieldCount; intField++)
{
   Console.WriteLine("Field #{0}", intField);
   Console.WriteLine("Name: {0}", rdr.GetName(intField));
   //显示字段序号为 intField 的字段名称
   Console.WriteLine(" .NETdatatype: {0}", rdr.GetFieldType(intField));
   //显示字段序号为 intField 的字段数据类型
   Console.WriteLine();
```

```
}
rdr.Close();
}
```

7.5.4　DataAdapter 对象

DataAdapter（数据适配器）对象在 ADO.NET 对象模型中是连接处理和断开连接处理的纽带，用于在数据源和数据集（DataSet）之间交换数据。可以使用 DataAdapter 对象将数据从数据库中取出，然后填充（Fill）到 DataSet（数据集）中去。DataAdapter 还可以获取 DataSet 中数据的更新，并将它们提交给数据库。参见图 7-7 和图 7-8。

DataAdapter 对象是在 Command 对象的基础上建立的对象，在需要存取访问时会连接数据库，DataAdapter 对象的属性分为如下两种情况。

（1）用来控制数据读取、插入、修改、删除或更新的属性，这些属性包括：

① DeleteCommand：取得或设置从数据源删除记录的 SQL 命令；

② InsertCommand：取得或设置从数据源新增记录的 SQL 命令；

③ SelectCommand：取得或设置从数据源查询记录的 SQL 命令；

④ UpdateCommand：取得或设置从数据源更新记录的 SQL 命令。

（2）用来控制与数据集之间通信的属性。当在数据集和数据源之间进行数据信息交互时，DataAdapter 对象会执行与之相关的*Command 对象来获取相关的操作信息，并且将它存入 DataSet 中，或是将存储在 DataSet 中的更改提交给数据库存储。

创建 DataAdapter 对象主要有两种方法：一种是在设计器中创建 DataAdapter 对象；另一种是在运行时直接通过代码创建 DataAdapter 对象。代码创建时可使用 DataAdapte 构造函数来创建 DataAdapter 对象。构造函数如下：

```
DataAdapter(CommandText,Connection);
```

其中，CommandText 是一个字符串，为 SQL 的 SELECT 语句命令串或存储过程名，将作为 DataAdapter 的 Command 属性。Connection 表示连接对象。假设 conn 是一个数据库连接对象，且已打开，下列一组语句构建了一个 DataAdapter 对象：

```
…
strSQL    string ="SELECT * FROM Students" ;// 定义 SQL 查询语句串
MyAdapter=New SqlDataAdapter(strSQL, conn) ;
//创建并初始化 SqlDataAdapter 对象，同时执行 SQL 语句
…
```

DataAdapter 对象常与 DataSet 对象配合使用，可以实现非连接方式下的数据操作。当使用 DataAdapter 对象执行完访问数据库的命令后，可用命令 DataSet 对象的 Fill()方法将结果集装入 DataSet，然后关闭连接，使用 DataSet 对象集合中的 DataTable 对象进行数据的增、删、改及查询。

【7-18】使用 DataAdapter 对象读取 student 数据库表的数据。

假设将数据显示页面命名为 DataAdapter_Ex01.aspx，则在后台 DataAdapter_Ex01.aspx.cs 文件中，添加命名空间定义及页面装入事件代码。

```
//导入命名空间
using System;
using System.Data;
using System.Data.SqlClient;
//页面装入事件处理代码
protected void Page_Load(object sender, System.EventArgs e)
{
      string sConnectionString = "";       //声明一个字符串
```

```
        sConnectionString = " Data Source=.;Initial Catalog=student;User ID=sa; ";
        //连接数据库字符串，连接到 student 数据库，用户名是 sa
        SqlConnection conn = new SqlConnection(sConnectionString);
        //创建 SqlConnection 数据库连接对象
        conn.Open();      //打开 Conn
        string sql = "Select * From student";        //定义 SQL 语句串
        SqlDataAdapter da = new SqlDataAdapter(sql, Conn);
        //创建并初始化 SqlDataAdapter 对象，同时执行 SQL 语句
        DataSet ds = new DataSet();     //声明并创建 DataSet 的一个实例 ds
        da.Fill(ds, "student");     //将 DataAdapter 检索的数据填充到数据集 ds
        conn.close();
    }
```

该例在前台页面装入时，建立与 student 数据库的连接，创建 DataAdapter 读取 student 表数据，da.Fill(ds, "student")语句将数据填入 DataSet 中。

7.5.5 DatatSet 对象

DataSet 对象类似于 ADO 中的 RecordSet 对象，不同的是它支持断开连接时的数据操作。DataSet 对象由许多数据表、记录和字段组成，是一个驻留内存的数据库。它可用于多种不同的数据源，包括基于 XML 的数据。

DataSet 对象模型如图 7-9 所示。一个 DataSet 对象可以由若干 Collection（集合）构成，而一个 Collection 又可以由若干对象组成。因此一个 DataSet 对象可以认为是 DataTableCollection、DataRelationCollection、ExtendedProperties 等对象的集合。

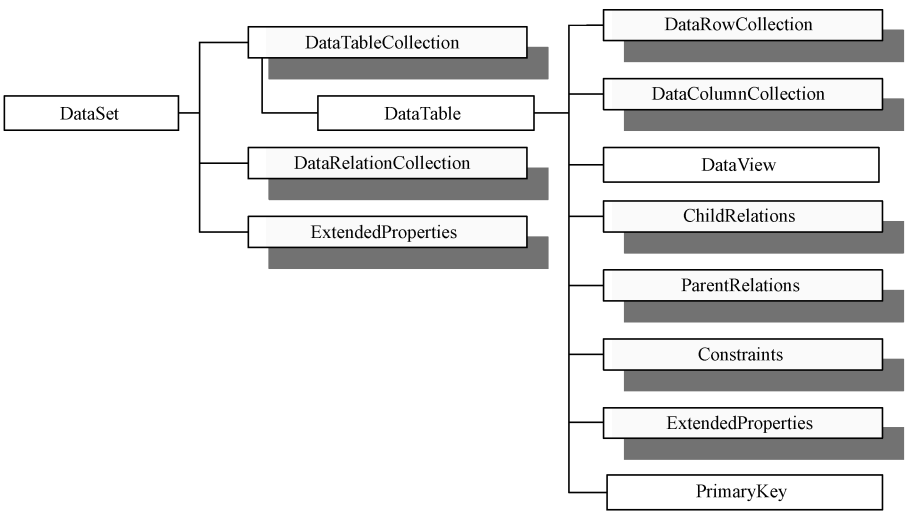


图 7-9　DataSet 对象模型

（1）DataTableCollection。它包含 DataSet 中所有的 DataTable 对象。DataTable 在 System.Data 命名空间中定义，是表示内存驻留数据的单个表。它是包含 DataColumnCollection 所表示的列和 Constrains 所表示的约束的集合，这些列和约束一起定义了数据表的结构。DataTable 还包含 DataRowCollection 所表示的行的集合，而 DataRowCollection 则包含表中的数据。

数据表中的 Constrains 集合包含零个或多个约束。在关系数据库中，约束用来维护数据库的完整性。ADO.NET 支持两种形式的数据约束：外键约束和唯一键约束。外键约束维护关系的完整性，唯一键约束维护数据的完整性，以确保表中不能有相同的行。

（2）DataRelationCollection。DataRelation 对象定义不同 DataTable 对象之间的关系，标识 DataSet 中两个表的匹配列，并可通过 DataRow 对象获取，关系类似于存在于关系数据库中的主键列和外键列之间的连接路径。

关系能够在 DataSet 中从一个表导航至另一个表。DataRelation 的基本元素为关系的名称、相关表的名称及每个表中的相关列。关系可以通过一个表的多个列来生成，方法是将一组 DataColumn 对象指定为键列。

（3）ExtendedProperties。DataSet、DataTable 和 DataColumn 都具有 ExtendedProperties 属性。它是一个 PropertyCollection，其中放置自定义的信息，如用于生成结果集的 SELECT 语句或表示数据生成时间的日期/时间戳。

DataSet 对象的创建可以使用 DataSet 对象构造函数，例如：

```
ds　DataSet = New DataSet();
```

DataSet 对象访问数据源时，需要通过 DataAdapter，参见图 7-8。DataSet 取得数据的方式有两种：一种是利用程序通过访问数据库取得；另一种是读取 XML 文件。

【例 7-19】通过读取数据库数据创建一个 DataSet。

假设名为 DataSet_Ex01.aspx 的网页需要将数据库表 student 的内容读入数据集对象 DataSet，则可在 DataSet_Ex01.aspx.cs 文件的事件处理程序 Page_Load()中添加如下相应代码：

```
string strconn = "Data Source=(local);Initial Catalog=student;User ID=sa;Password=";
string sql = "SELECT sno, sname,ssex,sage,sdept FROM student";
//定义 SQL 查询语句
SqlConnection Conn=new SqlConnection(strconn);
Conn.Open();
SqlDataAdapter da = new SqlDataAdapter(sql, conn));
//创建并初始化 SqlDataAdapter 对象，同时执行 SQL 语句
DataSet ds = new DataSet();          //声明并创建 DataSet 的一个实例 ds
da.Fill(ds, "student");
// DataAdapter 对象把 DataSet 和具体数据库联系起来，Fill()方法填充数据给 DataSet
Conn.Close();
... //这里可以在非连接方式下处理 DataSet 对象集合中的数据
```

7.5.6　DataTable 对象

DataTable 类是一个重要的.NET 框架类，代表一个单独的数据库表的集合，它存放于 DataSet 数据集中。在 ADO.NET 中，DataTable 是构成 DataSet 最主要的对象。DataTable 对象用于表示 DataSet 中的表，但该表并不是单独存在于 DataSet 中，在同一个 DataSet 对象中可以包含多个 DataTable 表。DataTable 对象由 DataColumns 集合和 DataRows 集合组成，通过数据控制控件或数据源对象，从数据库获取数据后，填充在 DataSet 数据集的各个 DataTable 对象中。当访问 DataTable 对象时，要注意它们是按条件区分大小写的。

1．创建 DataTable 对象

DataTable 对象的创建可以使用相应的 DataTable()构造函数，例如：

```
DataTable StuTable =New DataTable();
```

通过使用 Add()方法可以将对象添加到 DataSet 对象的 DataTables 集合中，DataTable 的 Add()方法主要有：

① Add()：使用默认名称创建一个新的 DataTable 对象，并将其添加到集合中；

② Add(DataTable)：将指定的 DataTable 添加到集合中；

③ Add(String)：使用指定名称创建一个 DataTable 对象，并将其添加到集合中；

④ Add(String,String)：使用指定名称创建一个 DataTable 对象，并将其添加到集合中。

也可以通过使用 DataAdapter 对象的 Fill()方法或 FilSchema()方法创建 DataTable 对象，或者使用 DataSet 的 ReadXml()、ReadXmlSchema()或 InferXmlSchema()方法从预定义的 XML 架构中创建。注意，将一个 DataTable 作为成员添加到一个 DataSet 的 Tables 集合后，不能再将其添加到任何其他 DataSet 表的集合中。

例如，要向 ds 数据集中添加一个表名为 course 的新表，可以使用如下代码：

```
dt = ds.Tables.Add("course");
dataAdapter.Fill(dt);          //将 DataAdapter 对象读取的数据填入表对象 dt 中
```

此外，还可以向新表中添加列，代码如下：

```
dt.Columns.Add("cno");
dt.Columns.Add("cname");
```

DataRow 和 DataColumn 对象是 DataTable 的主要组件，即 DataTable 的行与列，它们作为 DataTable 的两个集合元素存在。例如，ds.Tables["student"].row[2]["sno"]表示数据表 student 的第三行的 sno 列，通过 DataRow 和 DataColumn 就能确定其具体值。

通常使用 DataRow 对象及其属性和方法检索、插入、删除和更新 DataTable 中的值。

【例 7-20】创建 DataTable 对象存放 DataAdapter 读入的数据，并显示于页面中。

假设 DataTable.aspx 页面为显示页面，则在 DataTable.aspx.cs 文件的 Page_Load()中添加如下代码：

```
protected void Page_Load(object sender, System.EventArgs e)
{
    string sConnectionString = "";
    sConnectionString = " Data Source=.;Initial Catalog=student;User ID=sa; ";
    SqlConnection Conn = new SqlConnection(sConnectionString);
    Conn.Open();
    string sql = " Select cno AS  课程号,cname AS 课程名,ccredit AS  学分 From course ";
    SqlDataAdapter dataAdapter = new SqlDataAdapter(sql, Conn);
    DataTable dt = new DataTable();     //创建 DataTable 对象
    dataAdapter.Fill(dt); //用 fill 方法将 DataAdapter 的数据填充到 DataTable
        数据表对象中
    DataGrid1.DataSource = dt;
    DataGrid1.DataBind(); //将 DataTable 绑定到 DataGrid 控件
}
```

DataGrid1.DataSource = dt 和 DataGrid1.DataBind()两条语句将数据表 dt 绑定在页面的显示控件上，运行效果如图 7-10 所示。有关控件绑定的内容将在 7.6 节讨论。

图 7-10　使用 DataTable 运行效果

2．DataTable 对象的数据操作

对于 DataTable 对象，我们可以使用数据库表一样的的插入、删除、更新、检索等操作。

（1）读写 DataTable 对象中的数据

即读写 DataTable 中某条记录的字段值，需要使用 DataTable 对象的 Rows 集合，记录数从零开始。如某 DataSet 对象拥有表名为 mytable 的数据表，若 ds 是 DataSet 对象，把该表中第一条记录 name 字段的值改为“李明”，代码如下：

```
DataTable tbl=ds.Tables["mytable"];//按表名在 DataTableCollection 中获取 DataTable 对象
DataRow dr1=tbl.Row[0] ; //取第一行记录
```

```
dr1["name"]="李明"; //将 name 字段的值写为“ 李明 ”
```

（2）插入一条记录

在 DataTable 对象中插入一条记录，首先需要创建一个 DataRow 对象，然后在 DataTable 对象的 DataRowCollection 中添加该 DataRow 对象，代码如下：

```
DataRow newrow=ds.Tables["mytable"].NewRow();
//创建一个 DataRow 对象，且和 mytable 表具有相同的字段
newrow["name"]="李明"; //字段赋值
    ...
ds.Tables["mytable"].Rows.Add(newrow); //再添加一条新记录行
```

（3）删除一条记录

从数据表中删除一条记录，可以采用两种方法。一是使用 DataRowCollection 对象的 Remove()方法，首先在 DataRowCollection 对象中定位需要删除的记录，然后调用 Remove()方法。例如，要求删除表名为 mytable 的数据表中的第 4 条记录，代码如下：

```
DataTable tb=ds.Tables["mytable"];
DataRow dr = tb.Rows(3); //指定要删除的行
tb.Rows.Remove(dr); //删除 dr 代表的记录行
```

二是使用 DataRow 对象的 Delete()方法，代码如下：

```
dr.Delete;
```

（4）检索和排序

在 DataTable 对象中对数据进行检索和排序，需要使用 DataTable 对象的 Select()方法，获取 DataRow 对象的数组。DataTable 对象的 Select()方法使用方式有如下几种。

若 dt 是 DataTable 对象，则获取所有行的代码如下：

```
DataRow[] rows = dt.Select();
```

若要按条件检索，则代码如下：

```
DataRow[] rows = dt.Select("id>1401", "id DESC");
```

"id DESC"是指按 id 降序排列结果。

（5）数据统计

使用 DataTable 对象的 Compute()方法可以对数据进行求和、平均、计数等数据统计，该方法与 SQL Server 中使用 Sum()、Aver()、Count()等操作获得数据的统计结果类似。

例如，统计所有性别为女的学生的数量，若 ds 是 DataSet 对象，代码如下：

```
DataTable tb=ds.Tables["student"];
object n = table.Compute("count(id)", "sex =女");
```

7.5.7　DataView 对象

在 DataTable 中使用 SQL 语句的 Select()方法，可实现对数据的筛选与排序。虽然 Select()方法功能强大，使用灵活，但是由于 SQL 语句接受动态的查询条件，因此使用时效率并不高。并且数据表 Select()方法返回的是一个数据行数组，不能直接用于数据绑定。为了解决这个问题，ADO.NET 设计了 DataView，它提供一些处理数据的机制，其中包括：

① 排序：可以以升序或降序方式、按一列或多列进行排序；

② 筛选：可以用于对一列或多列表达式进行数据筛选；

③ 数据行版本过滤：可视的数据可根据数据行的版本进行过滤。

使用 DataView，可以按不同的排序顺序显示表中的数据，并且可以按行状态或基于筛选器表达式来筛选数据。DataView 提供基础 DataTable 中数据的动态视图，内容、排序和成员关系会实时反映其更改。这种机制不同于 DataTable 的 Select()方法，后者从表中按特定的筛选器或排序顺序返回 DataRow 数组，虽然其内容可反映对基础表的更改，但其成员关系和排序却保持静态。

DataView 的动态功能使其成为数据绑定应用程序的理想选择。

这里需要注意数据库视图与 DataView 的区别。数据库视图实际上是一个查询，在数据库中创建视图时，要返回视图数据查询。而 ADO.NET 中的 DataView 虽然可以进行数据的筛选、排序和查找，但是它并非 SQL 查询，因此不能作为表对待，也无法提供两个表的连接，无法看到数据表中的某一列。

【例 7-21】用 DataView 对象实现数据在页面上排序显示。显示效果如图 7-11 所示。

创建 DataView_Ex01.aspx 页面，添加一个 DataGrid 控件 DataGrid1；两个 Label 控件，Text 属性值分别为“课程选择”和“排序方式”；一个 DropDownList 控件，属性 AutoPostBack=true；两个 RadioButton 控件，Text 属性值分别设为“降序排列”和“升序排列”，属性 AutoPostBack=true，GroupName 为“RadioGroup1”。对应后台 C#程序代码如下：

```
protected void Page_Load(object sender, System.EventArgs e)
{
    string str = "Data Source=(local);Initial Catalog=student;User ID=sa;";
    if (!IsPostBack)  //判断是否首次加载页面
    {
    SqlConnection cn = new SqlConnection(str);          //创建连接数据库
    cn.Open();
    SqlCommand cmd = new SqlCommand();                  //创建 Command 命令
    cmd.Connection = cn;
    cmd.CommandText = "SELECT distinct cno FROM sc ";
    //定义 Select 命令
    SqlDataReader rdsdept = default(SqlDataReader);   //创建 SqlDataReader 对象
    rdsdept = cmd.ExecuteReader();   //执行命令，返回 Select 执行结果
    while (rdsdept.Read())
    {
        DropDownList1.Items.Add(rdsdept["cno"]);
            //将课程号添加到 DropDownList1 控件中
    }
    rdsdept.Close();
    cn.Close();
    }
    string strsql = "select * from sc";    //声明一个 SQL 命令字符串
    SqlDataAdapter da = new SqlDataAdapter(strsql, str);
    //创建并初始化 SqlDataAdapter 对象，同时执行 SQL 语句
    DataSet ds = new DataSet();    //声明并创建 DataSet 的一个实例 ds
    da.Fill(ds, "sc");      //填充数据集
    DataView dv = new DataView();    //声明并创建一个 DataView 实例
    dv.Table = ds.Tables["sc"];
    //将 DataSet 中的 sc 数据表作为 DataView 的数据源，这两句也可以用下面
        这个构造方法来实现
        DataView dv=new DataView(ds.Tables[“sc”]);
    dv.RowFilter = "cno=\'" + DropDownList1.Text + "\'";
    //根据 DropDownList 控件中的内容进行数据筛选
    if (RadioButton1.Checked == true)
    {
        dv.Sort = "grade desc";          //按 grade 降序排列
    }
    if (RadioButton2.Checked == true)
    {
        dv.Sort = "grade asc";   //按 grade 升序排列
```

```
            }
            DataGrid1.DataSource = dv;
            DataGrid1.DataBind();//将 DataView 作为 DataGrid 的数据源，绑定 DataGrid
            }
        }
```

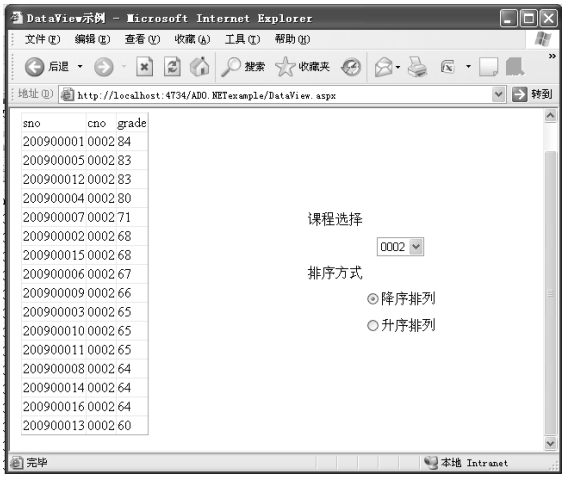


图 7-11　课程选择“ 0002 ”、排序方式选择“ 降序排列 ”的运行结果

7.6　数据源与 Web 控件的绑定

Web 页面支持多种控件，除了通用的 TextBox、CheckBox、ListBox、Label、Button 外，还包括 Repeater、DataList、DataGrid 等适用于多记录数据显示的控件。数据绑定是指将控件和数据源捆绑在一起，通过控件来显示或修改数据源。数据绑定技术将数据连接、SQL 命令设置、数据操作等功能集成为一体，不仅简化了数据库连接操作，也使访问数据源的代码量大大减少。数据绑定被定义成.NET 中控件体系结构的一部分。Windows 窗体和 Web 窗体都支持数据绑定。

可以绑定到的控件的数据源类型包括：DataColumn 对象、DataTable 对象、DataView 对象、DataSet 对象、DataViewManager 对象、数组或集合。

7.6.1　数据绑定方法

Web 窗体页中的数据绑定十分灵活，可以将控件的任何属性绑定到数据。例如，可以设置文本属性以在 TextBox、Label、Button、LinkButton 等控件中显示文本；将 CheckBox 控件绑定到布尔值以直接设置该控件的选中状态；通过将 Image 控件的 ImageURL 属性绑定到包含图形文件的 URL 或数据库列来设置 Image 控件的图形；通过绑定设置控件的颜色、字体或大小等。

Web 页面中某些控件只可显示单个值，如 Label、TextBox、CheckBox 等，而有些部件却可显示多个数据值，如 Repeater、DataList、DataGrid、ListBox 等。对应数据的绑定就有简单绑定和复杂绑定两种类型。每个控件，都有一个“ DataBindings ”属性，用于绑定数据源。绑定可以在设计控件时通过设计工具箱完成，也可通过编写代码在运行时完成。

1．设计时数据绑定

可以使用控件的 DataBindings 属性在设计时实现简单数据绑定，简单绑定的基本步骤如下。

（1）选择需绑定的控件，在属性窗口中单击 DataBinding，显示数据绑定对话框。

（2）在可绑定属性中，指定要绑定到数据源的属性。如 TextBox 控件的 Text 属性。

（3）选择简单绑定，指定数据源的数据成员，以便获取特定的数据。

对于大多数的控件，复杂绑定的设置是通过 DataSource 属性实现的，复杂绑定的基本步骤如下。

（1）在窗体中，选择该控件并显示属性窗口，然后单击 DataBinding，即可看到与控件对应的默认绑定属性。

（2）单击“ Advanced ”属性右边的“ … ”按钮，显示“ 格式设置和高级绑定 ”对话框，在此对话框中，选择要绑定的数据源和被绑定的属性。即将控件的 DataSource 属性设置为某个数据源，如 DataTable、DataView、DataSet 等。

（3）在控件的 DisplayMember 属性中设置数据源的列名。

归纳起来，设计时建立数据绑定的一般步骤为：选定控件，进行数据源配置，选定要绑定的数据源，连接数据源，设置数据检索 Select 语句。

【例 7-22】运用设计工具在设计时实现 DataList 控件的数据绑定。

图 7-12　加入 DataList 控件

在工具箱中选中“ 数据 ”栏中的 DataList 控件，添加到设计的页面中，此时会弹出“ DataList 任务 ”对话框，如图 7-12 所示。选择“ 自动套用格式 ”功能设置 DataList 控件外观显示的效果。“ 编辑模板 ”功能可用模板形式设置 DataList 控件的显示效果。

“ 选择数据源 ”栏选择“ 新建数据源 ”项进入“ 数据源配置向导 ”对话框，如图 7-13 所示。在“ 选择数据源类型 ”标签下，选择“ 应用程序从哪里获取数据？ ”项为“ 数据库 ”，“ 为数据源指定 ID ”项输入 SqlDataSource1，单击“ 确定 ”按钮，建立新数据源，连接数据源并测试。

图 7-13　选择数据源

测试连接成功，单击“ 确定 ”按钮，就可以把数据库连接字符串添加到“ 配置数据源 ”对话框中。单击“ 下一步 ”按钮，弹出“ 将连接字符串保存到应用程序配置文件中 ”对话框，将字符串写入配置文件中，其好处是可以在整个项目中随时随地地调用配置文件中与数据库连接的字符串，而不必每次都重新设置。

完成数据源配置操作后，进入“ 配置数据源 ”对话框的“ 配置 Select 语句 ”标签下，设置获取控件数据的 Select 语句。配置 Select 语句有两种方式。一种是指定自定义 SQL 语句或存储过程，另一种是指定来自表或视图的列，如图 7-14 所示。

图 7-14　配置 Select 语句

本例选择“指定来自表或视图的列”，选择 course 表，选中表中的 cno、cname、ccredit 三个字段，“Select 语句”栏中显示所设置的 SQL 命令语句为“SELECT [cno], [cname], [ccredit] FROM [course]”。

也可以选择“指定自定义 SQL 语句或存储过程”，单击“下一步”按钮，在弹出的“自定义 SQL 语句”对话框中输入 SQL 语句。

2．运行时数据绑定

在某些情况下，设计时指定绑定数据是不切实际的。如果需要绑定的数据源在设计时不具有实例，则需要在运行时执行数据绑定。需要在运行时进行数据绑定的示例包括：

（1）将控件绑定到通过执行 SQL 语句或存储过程返回的数据；

（2）绑定到运行时未实例化的任何类型的对象。

运行时的数据绑定通常通过代码设置控件的 DataSource 属性和 DataBind()方法来实现。例如，例 7-21 中的下列语句：

```
DataView dv = new DataView(ds.Tables["sc"]);
DataGrid1.DataSource = dv;
DataGrid1.DataBind();     //将 DataGrid1 和 dv 对象绑定
```

在下面的控件绑定示例中，我们将会看到两种绑定方式的运用。

7.6.2　Repeater Web 控件绑定

Repeater 控件是一种简单实用的 Web 控件，它通过列表来显示数据项。Repeater 控件可以处理一些事件，但也有所限制，它不支持对数据的选择、编辑功能，对于复杂的处理和编辑功能，应使用 DataList 或 DataGrid 控件。

【例 7-23】用设计工具实现 Repeater 控件的数据绑定，显示从 course 中检索出的数据。

建立名为 Repeater.aspx 的网页，在“工具箱”的“数据”栏中直接将 Repeater 控件拖放到“设计”视图的页面中，并进行数据绑定。将<ItemTemplate>元素作为 Repeater 控件的子集添加到页面中，向 ItemTemplate 添加 HTML 编码及其他所需 Web 服务器控件或 HTML 服务器控件。运行 Repeater.aspx，页面显示效果如图 7-15 所示。

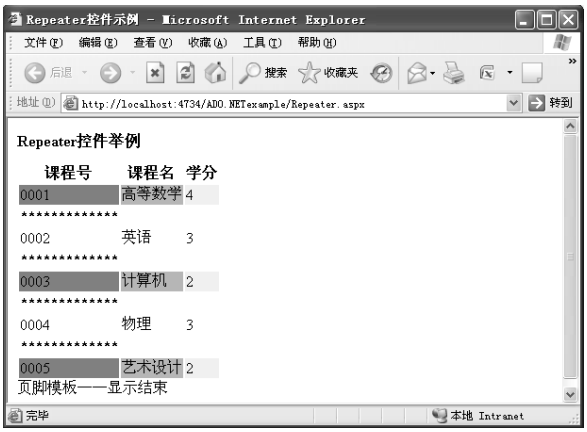


图 7-15　Repeater.aspx 页面显示效果

7.6.3　DataList 控件绑定

DataList 控件是 ASP.NET 中与列表紧密相关的数据绑定控件，它主要用于创建和显示模板化的列表数据。DataList 控件使用表格显示数据源控件获取的记录数据。在 DataList 控件中不仅支持显示、分页、编辑等功能，而且还支持强大的模板设计功能。它也允许通过手动设置方式与数据源建立连接，这样可以减少代码编写量。例 7-24 将用代码实现运行时 DataList 控件的数据绑定。

【例 7-24】编写代码完成运行时对 DataList 控件的数据绑定。

创建名为 DataList.aspx 的网页，在“工具箱”的“数据”工具栏中找到“DataList”控件，拖放到页面上。在 DataList.aspx.c#文件中添加如下代码实现绑定：

```
protected void Page_Load(object sender, System.EventArgs e)
{
    using System.Data;
    using System.Data.SqlClient;
    string sConnectionString = "";
    sConnectionString = " Data Source=.;Initial Catalog= student;User ID=sa; Password=";
    SqlConnection Conn = new SqlConnection(sConnectionString);
    Conn.Open();
    string sql = "SELECT cno, cname, ccredit FROM course";
    SqlDataAdapter infor = new SqlDataAdapter(sql, Conn);
    DataSet ds = new DataSet();
    infor.Fill(ds, "infor");
    //使用 SqlDataAdapter 的 Fill()方法填充 DataSet
    DataList1.DataSource = ds.Tables["infor"];
    DataList1.DataBind();  // DataList1 控件的数据绑定
}
```

本方法先将 SQL Server 数据库中取出的数据通过 SqlDataAdapter 的 Fill()方法填充到数据集中，然后将数据集中的数据作为 DataList 控件的数据源，最后在 DataList 控件上显示数据信息。

7.6.4　DataGrid 控件绑定

DataGrid 是 Web 服务器控件，以表格方形布局显示数据，提供丰富的数据访问功能。DataGrid 为数据源中的每个字段生成一个 BoundColumn（AutoGenerateColumns=true 时）。数据中的每个字段按照在数据中出现的顺序呈现在单独的列中，字段名显示在网格表的列标题上，值呈现在文本域中。

在默认情况下，DataGrid 以只读模式显示数据，但是它也能在运行时自动在可编辑控件中显示数据。DataGrid 控件可通过“属性生成器”对话框创建“选择”、“编辑”、“更新”和“取消”按钮及编程结构。另外，DataGrid 支持分页功能，也可以使用控件的自定义导航功能通过控制发送到客户端浏览器的数量来提高性能。DataGrid 综合了 Repeater 控件与 DataList 控件的特点，使数据的显示和编辑更加方便。

【例 7-25】DataGrid 控件应用示例。创建名为 DataGrid.aspx 的网页，要求具有对数据库数据的编辑、更新、取消和分页等功能，页面效果如图 7-16 所示。

图 7-16　DataGrid 控件应用示例

首先在页面上添加一个 DataGrid 控件 DataGrid1，在“DataGrid 任务”命令框中，选择“自动套用格式”的“彩色型”显示 DataGrid 中的数据；选择“属性生成器”，打开如图 7-17 所示的对话框。单击“列”项，在“可用列”栏添加 4 个模板列到“选定的列”，页眉文本分别设置为“学号”、“姓名”、“性别”和“所在学院”；添加“编辑、更新、取消”按钮列到选定列，设置页眉文本为“编辑”；添加“选择”按钮列到选定列，设置页眉文本为“选择”，其他属性采用默认值。

图 7-17　属性生成器中“列”的设置

单击“属性生成器”的“分页”项，选中“允许分页”，页面大小设置为“6”行，导航设置为“上一页”、“下一页”按钮。

分别在“姓名”、“性别”和“所在学院”的编辑模板列中放入一个 TextBox 控件，以便更新时填入新数据。

在 DataGrid.aspx.cs 文件中添加如下相应编码：

```
private void DataGridDataBind()    //自定义函数，进行数据库访问和绑定
{
```

```
        string sConnectionString = "";
        sConnectionString = " Data Source=.;Initial Catalog= student;User ID=sa;";
        SqlConnection Conn = new SqlConnection(sConnectionString);
        SqlDataAdapter Adapter = new SqlDataAdapter("SELECT sno,sname,ssex,
            sdept FROM student",
        Conn);
        Conn.Open();
        DataSet ds = new DataSet();          //声明并创建 DataSet 的一个实例 ds
        try
        {
            Adapter.Fill(ds, "testTable");
            //将 DataSet 中的数据信息绑定到 DataGrid 控件上
            DataGrid1.DataSource = ds.Tables["testTable"].DefaultView;
            DataGrid1.DataBind();
        }
        catch (Exception ex)
        {
            Response.Write(ex.ToString());       //抛出异常信息
        }
        finally
        {
            Conn.Close();
        }
    }
    protected void Page_Load(object sender, System.EventArgs e)// Page_Load()事件
    {
        if (!IsPostBack)
            DataGridDataBind();//调用 DataGrid1.DataBind()函数
    }
    //添加 PageIndexChanged()事件处理程序的代码
    Protected void DataGrid1_PageIndexChanged
        (object source, System.Web.UI.WebControls.DataGridPageChangedEventArgs e)
    {
        DataGrid1.CurrentPageIndex = e.NewPageIndex;    //把当前行变成可编辑状态
        DataGrid1.DataBind();
    }
```

通常需要为 DataGrid 控件分别添加 EditCommand()、UpdateCommand()、CancelCommand()事件处理程序，这里不再详述。

7.6.5 GridView 控件绑定

在 ADO.NET 中，GridView 控件是 Web 程序中的表格数据绑定控件，它是 DataGrid 控件的后继控件。使用 DataGrid 控件时，数据的排序、分页和编辑需要附加的编码。GridView 控件则无须编写任何代码即可添加排序、分页和编辑功能，可以通过在控件上设置属性来自动完成这些任务。

GridView 控件的绑定过程与其他控件类似，这里不再赘述。在“GridView 任务”对话框中分别选择“启用分页”、“启用排序”和“启用选定内容”等复选框，无须编码就可以自动完成分页、排序等功能，如图 7-18 所示。使用“启动分页”命令时，在 GridView 控件的属性中可以设置每页显示的数据记录数。默认情况下，PageSize 的值是 10，也可以根据需要重新设置。

图 7-18　GridView 控件定义

7.7　ADO.NET 数据库访问示例——学生成绩查询与修改

【例 7-26】对 student 数据库中的 course 数据表进行数据查询并显示在 DataList 控件中，同时实现数据的编辑、删除、更新和取消操作。

（1）启动 Visual Studio 环境，新建一个 ASP.NET 网站，将其保存在 ADO.NETexample 文件夹中，新建名为 AdoNet_Example.aspx 的网页文件。

（2）在 AdoNet_Example.aspx 的设计视图中，添加 HTML 表格、1 个 DataList 控件、3 个 Label 控件、3 个 TextBox 控件和 4 个 Button 控件，控件的 ID 和属性设置如表 7-2 所示。

表 7-2　控件及其属性设置

控　　件	属 性 设 置
Label 控件	ID 属性分别为 cnoLabel、cnameLabel、ccreditLabel
TextBox 控件	ID 属性为默认值
Button 控件	ID 属性为默认值，Text 属性依次为编辑、删除、更新和取消；CommandName 属性依次为 edit、delete、update 和 cancel

（3）右键单击 DataList 控件，在“ DataList 任务 ”对话框中选择“ 自动套用格式 ”的“ 简明型 ”格式。

（4）在“ DataList 任务 ”对话框中选择“ 编辑模板 ”，在“ 项模板 ”的“ ItemTemplate ”中将字段 cno、cname 和 ccredit 改成“ 课程号 ”、“ 课程名 ”和“ 学分 ”，并添加 2 个 Button 按钮“ 编辑 ”和“ 删除 ”；在“ 项模板 ”的“ EditItemplate ”中添加 3 个 TextBox 控件和“ 更新 ”和“ 取消 ”2 个按钮，如图 7-19 所示。在“ 页眉页脚模板 ”中分别添加页眉和页脚标题。

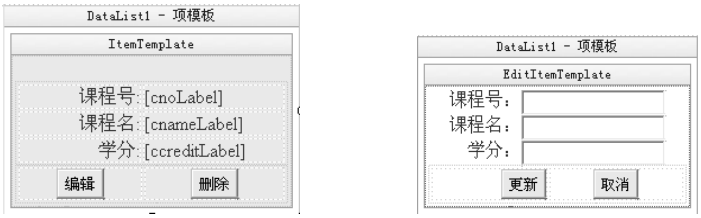


图 7-19　ItemTemplate 和 EditItemplate 设置

（5）在 AdoNet_Example.aspx.cs 文件中添加操作数据库要用到的命名空间：

```
using System.Data;
using System.Data.SqlClient;
```

（6）在 AdoNet_Example.aspx.cs 文件中自定义函数 DataListDataBind()，进行数据库访问和绑定，添加代码如下：

```
private void DataList1.DataBind()
{
    string sConnectionString = "";
    sConnectionString = " Data Source=.;Initial Catalog= student;User ID=sa;";
    SqlConnection Conn = new SqlConnection(sConnectionString);
    SqlDataAdapter Adapter = new SqlDataAdapter("SELECT * FROM course", Conn);
    Conn.Open();    //打开 Conn
    DataSet ds = new DataSet();        //创建 DataSet 对象
    try
    {
        //用 DataAdapter 的 Fill 方法填充数据集
        Adapter.Fill(ds, "testTable");
        //将数据集中的信息绑定到 DataList 控件中
        DataList1.DataSource = ds.Tables["testTable"].DefaultView;
        DataList1.DataBind();
    }
    catch (Exception ex) //捕获异常，输出异常信息
    {
        Response.Write(ex.ToString());
    }
    finally
    {
        Conn.Close(); //关闭 conn
    }
}
```

（7）在 AdoNet_Example.aspx.cs 文件的 Page_Load()事件中调用 DataList.DataBind()函数，添加代码如下：

```
protected void Page_Load(object sender, System.EventArgs e)
{
    if (!IsPostBack)
    DataList1.DataBind(); //页面首次打开时执行该语句
}
```

（8）为 DataList 控件分别添加 EditCommand()、UpdateCommand()、CancelCommand()和 DeleteCommand()事件处理程序代码：

```
Protected void DataList1_EditCommand(Object source,   DataListCommandEventArgs e)
{
    DataList1.EditItemIndex = e.Item.ItemIndex;   //把当前页变成可编辑状态
    DataList1.DataBind();
}
//EditCommand 事件处理程序中，由被单击的 Button 的索引来确定 EditItemTemplate
    要编辑的 EditItemIndex 的属性值，并将数据源绑定。
    Protected void DataList1_UpdateCommand(Object source ,
        DataListCommandEventArgs e)
    {
        Int ID = DataList1.DataKeys(e.Item.ItemIndex) ;
    TextBox newname = e.Item.FindControl("cname");
    String strUpt = "update course set cname ='" + newname.Text + "' where cno='" +
        ID.ToString() + "'"
    //定义一个更新命令字符串
    String   sConnectionString = " Data Source=.;Initial   Catalog= student;User ID=sa; "
    SqlConnection Conn = New SqlConnection(sConnectionString);
    SqlCommand cmd = New SqlCommand(strUpt, Conn);        //创建一个命令对象
    Conn.Open()
    Try
```

```
            { cmd.ExecuteNonQuery();     //执行 SQL 语句并返回影响的行数
            DataList1.EditItemIndex = -1 ; //放弃编辑
            DataList1.DataBind();   //绑定数据源
            }
            catch (Exception ex) //捕获异常，输出异常信息
            {
                Response.Write(ex.ToString());
            }
            finally
            {
            Conn.Close(); //关闭 conn
            }
    }
```

说明：UpdateCommand()事件处理程序中，使用 Item 事件参数对象的 Control.FindControl()方法读取当前项中被编辑的控件值，如 Label 和 TextBox 控件的 Text 值，并将更新结果写回到数据源。更新数据源后，通过 EditItemIndex 属性设置为-1，切换出编辑模式，然后重新将数据源绑定。

```
    protected void DataList1_CancelCommand(object source, DataListCommandEventArgs e)
    {
        DataList1.EditItemIndex = -1;        //放弃编辑
        DataList1.DataBind();
    }
```

说明：CancelCommand()事件处理程序允许用户在不保存更新的情况下退出编辑模式，只需将 EditItemIndex 属性设置为-1，然后重新将数据源绑定即可。

```
    protected void DataList1_DeleteCommand(object source, DataListCommandEventArgs e)
    {
        string kch = System.Convert.ToString(DataList1.DataKeys
            (e.Item.ItemIndex).ToString());
        //获取 id
        string sql = "delete from student where cno=\'" + kch + "\'";
        //定义一个删除命令字符串
        string sConnectionString = " Data Source=.;Initial Catalog= student;User ID=sa; ";
        SqlConnection Conn = new SqlConnection(sConnectionString);
        SqlCommand cmd = new SqlCommand(sql, Conn);
        Conn.Open();
        cmd.ExecuteNonQuery();        //执行 SQL 语句
        DataList1.DataBind();
    }
```

（9）将代码所在的页面设为起始页面，页面运行效果如图 7-20 所示。

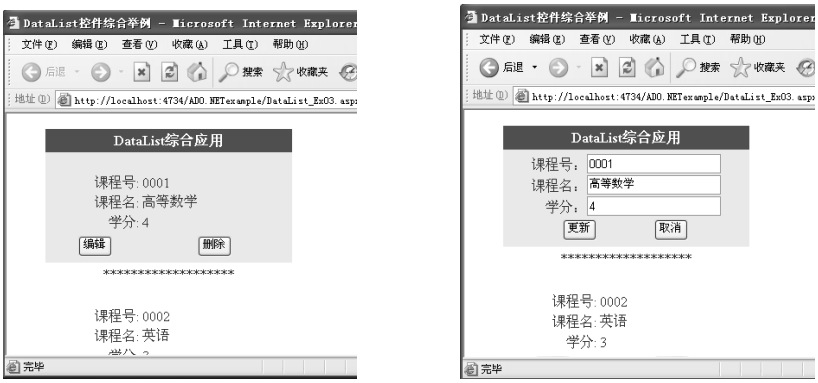


图 7-20　页面运行和单击“ 编辑 ”按钮后的页面

ADO.NET 的对象模型提供丰富的功能支持，由于篇幅的限制，只能简单介绍对象应用和数据库访问机制，对象的属性和方法、控件绑定的具体应用还需参考相关的技术手册。

本 章 小 结

本章着重介绍了基于.NET 框架的 Web 数据库访问技术，即利用 ADO.NET 实现对数据库的操作。

ADO.NET 具有四个数据提供者对象 Connection、DataReader、Command、DataAdapter 和一个用户对象 DataSet。Connection 对象提供与数据源的连接；Command 对象对数据源执行数据库命令；DataReader 从数据源中读取只读数据流；DataAdapter 提供连接 DataSet 对象和数据源的桥梁，它使用 Command 对象在数据源中执行 SQL 命令，以便将数据加载到 DataSet 中，并使 DataSet 中数据的更改与数据源保持一致；DataSet 对象用于存储从数据源获得的数据。

ADO.NET 提供两种数据库操作方式，连接模式和非连接模式。本章通过示例介绍了如何利用 ADO.NET 的数据库组件提供的对象读写数据库，以及如何运用 Web 的数据控件方便地操作数据库利用。

习 题 7

7.1 简述 ODBC 的主要功能。

7.2 简述 Windows 环境下，系统数据源、用户数据源和文件数据源的区别。试在一个 Windows 系统中分别建立一个 Microsoft Access 和一个 SQL Server 驱动的数据源。

7.3 现有两个数据库表：students 表记录学生的一般数据，其中字段 StudId 为学号，StudName 记录学生姓名；score 表记录学生的成绩，其中 Computer 字段和 English 字段分别记录 StudId 所表示的学生的计算机和英语课程的成绩。试写出下列数据库操作的 SQL 语句。

① 从成绩表中查询张强同学的计算机和英语成绩。

② 在表中加入一个学生的记录，其学号为 12010145，姓名为李雨，计算机成绩为 91，英语成绩为 82。

③ 将所有英语成绩高于 80 分（含 80 分）的同学的信息检索出来放入一个新表中，新表名为 highscore，包含的字段有学生的学号 StudId、姓名 StudName、英语成绩 English。

7.4 描述 ADO.NET 对象模型，说明对象之间有怎样的关系。

7.5 简述 ADO.NET 的两种数据库访问模式有何区别。

上机实验 7

实验 7.1 用 ADO.NET 技术设计并实现一个会员注册功能页面。

【目的】掌握 Web 页面与数据库间的数据交换方式。

【内容】为《Web 程序设计》课程网站设计并实现一个用户注册功能页面。注册界面如图 7-21 所示。

【步骤】

（1）建立一个用户数据库表，记录图 7-21 所示的信息。

（2）为步骤（1）中建立的数据库表配置一个系统数据源。

（3）建立一个 Web 页面文件，编程实现将图 7-21 所示的注册信息写入用户数据库表中。

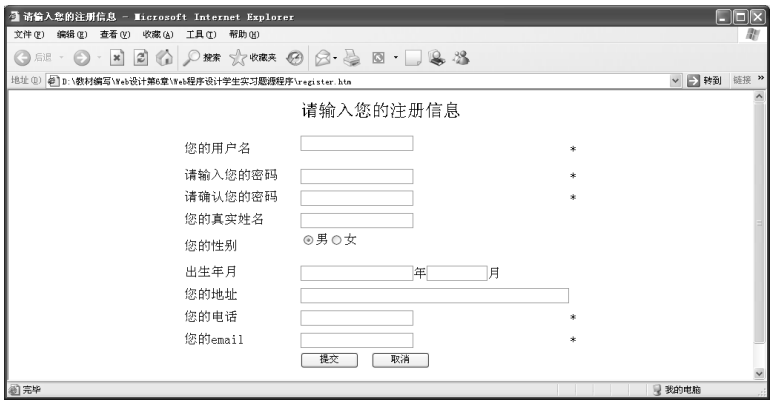


图 7-21　网站用户注册页面

实验 7.2　用 ADO.NET 对象和控件的数据绑定方法实现在 Web 页面上操纵数据库数据。

【目的】掌握数据库控件的数据绑定方法。

【内容】选择适当的 ADO.NET 控件，如 DataGrid 或 GridView，用数据绑定方法实现从 score 表中取出学号、姓名和成绩，并以表格形式显示在 Web 页面上，可实现编辑和删除功能。

【步骤】

（1）创建一个数据库表 score，包含学号、姓名和成绩字段。

（2）创建一个 ASP.NET 网页，加入可绑定数据库数据的表格控件。

（3）用 C#语言对编辑和删除事件编码，实现对数据库数据的更新操作。

第 8 章　ASP.NET 综合应用实例

8.1　实例 1——基于数据库的 BBS 论坛管理

本系统运用了 ADO.NET 数据访问对象，实现了基于 Web 的论坛帖子的增加、删除、修改、查询等简单的管理功能，其中包括管理员登录、发布、查询和维护帖子等功能。

8.1.1　功能设计

管理员以 admin/admin 用户身份/密码登录后，进入论坛主帖查询页面，分页显示所有主帖的标题；在查询页面中可以对某个帖子进行修改、删除操作，也可以查看帖子的详细信息；另外，还可以发布新帖子。

8.1.2　数据库设计

本实例选用 Access 数据库，所使用的数据库名为 MyBBS_Data.mdb，共有两张表。

（1）用户表 User。

用户表存储用户的信息，其中包括管理员用户，其结构如表 8-1 所示。

表 8-1　User 数据表结构

字 段 名	数据类型	可否为空	说　明
UserID	数值	否	用户唯一标识，主键，自动增量
UserLoginName	文本	否	登录名
UserName	文本	否	用户名
Password	文本	否	密码
Address	文本	是	住址
Homepage	文本	是	个人主页
Email	文本	是	邮箱地址

（2）主帖表 Topic。

主帖表存储用户发布的主帖信息，其结构如表 8-2 所示。

表 8-2　Topic 数据表结构

字 段 名	数据类型	可否为空	说　明
TopicID	数值	否	主帖唯一标识，主键，自动增量
UserLoginName	文本	否	发帖者登录名
TopicTitle	文本	否	主帖标题
TopicContent	备注	否	帖子内容
CreateTime	日期时间	是	发帖时间
IP	文本	是	发布机器 IP

8.1.3　界面设计

（1）登录页面设计：为整齐美观，在窗体中放入一个 3 行 1 列的 HTML 表格，在表格中合适

的位置输入“登录名”和“密码”文字，放置一个输入用户名的文本框、一个密码输入框、一个登录按钮。界面如图 8-1 所示。

（2）主帖查询页面设计：利用 GridView 控件分页显示主帖列表，并在 GridView 控件中添加修改、删除按钮和显示详细信息的超链接，另外在页面下方添加一个发表新帖的超链接。界面如图 8-2 所示。

图 8-1　登录窗体

帖子列表

编号	用户	标题	发表时间	修改	详细信息	删除
16	admin	<hi>	2008-11-15 16:15:47	修改	详细信息	删除
15	admin	再发个帖子	2008-11-15 16:14:47	修改	详细信息	删除
14	admin	发个帖子看看	2008-11-15 16:13:55	修改	详细信息	删除
13	admin	发个帖子看看	2008-11-15 16:13:13	修改	详细信息	删除
12	admin	MYBBS开始测试	2008-11-15 16:12:26	修改	详细信息	删除

1 2

查询结果（第1页 共2页） 发表新帖>>

图 8-2　主帖查询界面

（3）帖子详细信息页面设计：利用 Label 控件分别显示帖子标题、发帖人、发帖时间、帖子内容等信息。界面如图 8-3 所示。

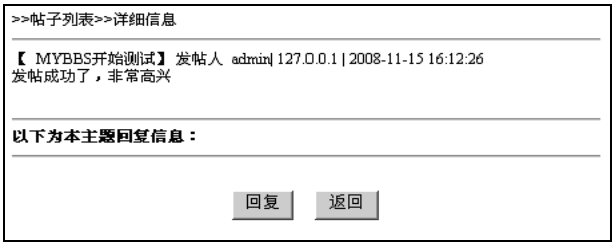


图 8-3　帖子详细信息界面

（4）发布帖子页面设计：利用文本框控件分别输入帖子标题、帖子内容等信息。界面如图 8-4 所示。

图 8-4　发布帖子界面

（5）修改帖子页面设计：与发布帖子界面类似。界面如图 8-5 所示。

图 8-5　修改帖子界面

8.1.4　关键技术

（1）GridView 定制：在 GridView 控件中，表格字段列均采用数据绑定列 BoundField 模板来定制，删除、修改采用数据绑定列 ButtonField 模板，详细信息使用数据绑定列 HyperLinkField 模板定制。

（2）分页：将 GridView 的 AllowPaging 属性设置为 True 为其启用分页功能，由于 GridView 的数据源在设计期间未绑定任何数据源控件，因此分页功能的代码需要手工编写。在 GridView 控件的 PageIndexChanging()事件处理程序中，通过设置其 PageIndex 属性值为新的页索引号来实现，新页索引号通过事件参数 GridViewPageEventArgs 的 NewPageIndex 属性值获得。

（3）数据库访问编程：删除、修改采用按钮数据绑定列定制，当用户单击删除、修改按钮时将触发 RowCommand()事件，因此在该事件处理代码中实现删除、修改操作。编程时利用 OleDbCommand 对象执行 Insert 和 Update 命令来实现相关功能。查询显示主帖列表是利用 OleDbDataAdapter、dataset 来实现的。

8.1.5　实现过程

（1）在 Visual Studio 网站 App_Data 文件夹中创建 Access 数据库 MyBBS_Data.mdb，建立用户表 User 和主帖表 Topic。配置数据库连接字符串，保存在 web.config 配置文件中。

（2）在 Visual Studio 网站中新建 manage 文件夹，在其中新建登录网页 Login.aspx、主帖查询页面 TopicList.aspx、帖子详细信息页面 TopicDetail.aspx、发布帖子页面 TopicAdd.aspx、修改帖子页面 TopicUpdate.aspx。

（3）分别编写网页的功能代码。

8.1.6　主要程序代码

（1）登录页 Login.aspx.cs 部分代码如下：

```
protected void ButtonLogin_Click(object sender, EventArgs e)
{       //获取用户在页面上的输入
        string userLoginName = TextBoxLoginName.Text.Trim();        //用户登录名
        string userPassword = TextBoxPassword.Text.Trim();          //密码
        OleDbDataReader dr; //  新建 DataReader 对象
        //  新建数据库连接 conn，连接到 Access 数据库
```

```
            System.Data.OleDb.OleDBConnection conn = new OleDbConnection();
            conn.ConnectionString =
                ConfigurationManager.ConnectionStrings["MyBBSConnectionString"].ConnectionString;
            OleDbCommand cmd = new OleDbCommand();                    // 新建 Command 对象
            cmd.Connection = conn;
            cmd.CommandText =
                    "SELECT * FROM [User] WHERE UserLoginName =@UserLoginName ";
            cmd.CommandType = CommandType.Text;
            // 添加查询参数对象,并给参数赋值
            cmd.Parameters.AddWithValue("@UserLoginName", userLoginName);
            try    // 打开 conn 连接，检索 User 表的 Password 字段
            {
                conn.Open();                          // 打开数据库连接
                dr = cmd.ExecuteReader();             // 将检索的记录行填充到 DataReader 对象中
                if (dr.Read())                        //如果用户存在
                {     // 如果密码正确，转入留言列表页面
                    if (dr.GetString(3) == userPassword)
                    {     // 使用 Session 来保存用户登录名信息
                        Session.Add("login_name", userLoginName);
                        Response.Redirect("TopicList.aspx");
                    }
                    else  //如果密码错误，给出提示
                    {
                        Response.Write("<Script Language=JavaScript>alert
                            (\"密码错误，请重新输入密码！\")</Script>");
                    }
                }
                else   //如果用户不存在
                {
                    Response.Write("<Script Language=JavaScript>alert(\"对不起，用户不存在！
                        \")</Script>");
                    }
                    dr.Close();   //关闭 DataReader 对象
            }
            catch (OleDbException oledbException)
            {     Response.Write(oledbException.Message);     }      // 显示连接异常信息
                finally
            {    // 如果连接打开则关闭连接
                    if (conn.State == ConnectionState.Open)
                    conn.Close();
                }
        }
```

（2）主帖查询页面 TopicList.aspx.cs 部分代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{
    if (!CheckUser())        // 如果用户未登录，强制转到登录页
            Response.Redirect("Login.aspx");
    if (!this.IsPostBack)    // 如果首次加载，调用 InitData()
            InitData();
}
private bool CheckUser()     //   验证用户是否登录
{  // 如果没有登录，提示用户登录
```

```
        if (Session["login_name"] == null)
    {
            Response.Write("<Script Language=JavaScript>alert('请登录！');</Script>");
            return false;
        }
        return true;
}
private void InitData()                                    // 按时间降序，读取帖子数据
{   // 新建数据库连接 conn，连接到 Access 数据库
    System.Data.OleDb.OleDbConnection conn = new OleDbConnection();
    conn.ConnectionString =
            ConfigurationManager.ConnectionStrings["MyBBSConnectionString"].ConnectionString;
    DataSet ds = new DataSet();                            // 新建 DataSet 对象
    // 新建 DataAdapter 对象，打开 conn 连接，检索 Topic 表的所有字段
    OleDbDataAdapter da =
new OleDbDataAdapter("SELECT * FROM [Topic] ORDER BY CreateTime DESC", conn);
    conn.Open();                                           // 打开数据库连接
    da.Fill(ds);                                           // 将检索的记录行填充到 DataSet 对象 ds 中
    conn.Close();                                          // 关闭数据库连接
    GV.DataSource = ds;
    GV.DataBind();
    LabelPages.Text = "查询结果（第" + (GV.PageIndex + 1).ToString() + "页 共" +
            GV.PageCount.ToString() + "页）";
}
private void deleteData(int topic_Id)                      // 删除帖子
{   // 新建数据库连接 conn，连接到 Access 数据库
    System.Data. OleDb. OleDbConnection conn = new OleDbConnection();
    conn.ConnectionString =
        ConfigurationManager.ConnectionStrings["MyBBSConnectionString"].ConnectionString;
    OleDbCommand cmd = new OleDbCommand();     // 新建 Command 对象
    cmd.Connection = conn;
    cmd.CommandText = "DELETE FROM [Topic] WHERE TopicID=@TopicID";
    cmd.CommandType = CommandType.Text;
    // 添加查询参数对象,并给参数赋值
    cmd.Parameters.AddWithValue("@TopicID ", topic_Id);
    try
    {
            conn.Open();                                   // 打开数据库连接
            cmd.ExecuteNonQuery();                         //将添加记录
            Response.Redirect("TopicList.aspx");
    }
    catch (OleDbException oledbException)
    {       Response.Write(oledbException.Message);    }      // 显示连接异常信息
    finally
    {       // 如果连接打开则关闭连接
            if (conn.State == ConnectionState.Open)
             conn.Close();
    }
}
protected void GV_RowCommand(object sender, GridViewCommandEventArgs e)
{
    int index = Convert.ToInt32(e.CommandArgument);        //待处理的行下标
```

```
        int topicId = -1;
        // 根据用户单击的按钮，执行不同程序
          switch (e.CommandName)
        {    //修改
              case "Update":
                    topicId = Convert.ToInt32(GV.Rows[index].Cells[0].Text);
                    Response.Redirect("TopicUpdate.aspx?topic_id=" + topicId);
                    break;
              //删除
              case "Delete":
                    topicId = Convert.ToInt32(GV.Rows[index].Cells[0].Text);
                    deleteData(topicId);
                    InitData();
                    break;
              default:
                    break;
        }
    }
    protected void GV_PageIndexChanging(object sender, GridViewPageEventArgs e)
    {    // 分页
        GV.PageIndex = e.NewPageIndex;
        InitData();                                                    // 刷新列表，显示新页
    }
```

上面的代码中，定义了一个 deleteData()方法用于删除指定 TopicID 的帖子。当用户单击删除按钮时将调用该方法，向其传递帖子的 TopicID 参数来删除指定帖子。

（3）帖子详细信息页面 TopicDetail.aspx.cs 部分代码如下：

```
    protected void Page_Load(object sender, EventArgs e)
    {    // 如果首次加载，调用 InitData()
        if (!this.IsPostBack)
              InitData();
    }
    protected void ButtonReply_Click(object sender, EventArgs e)
    {     }
    protected void ButtonBack_Click(object sender, EventArgs e)
    {      Response.Redirect("TopicList.aspx");     }
    private void InitData()
    {      // 获取链接传递的参数值
        int topicID = Convert.ToInt32(Request.QueryString["topic_id"]);
        // 新建数据库连接 conn，连接到 Access 数据库
            System.Data. OleDb. OleDbConnection conn = new OleDbConnection();
        conn.ConnectionString =
            ConfigurationManager.ConnectionStrings["MyBBSConnectionString"].ConnectionString;
        OleDbDataReader dr;                                            // 新建 DataReader 对象
        OleDbCommand cmd = new OleDbCommand();
        cmd.Connection = conn;
        cmd.CommandText = "SELECT * FROM [Topic] WHERE TopicID=@TopicID";
        cmd.CommandType = CommandType.Text;
        // 添加查询参数对象,并给参数赋值
          cmd.Parameters.AddWithValue("@TopicID ", topicID);
        try
        {
              conn.Open();                                    // 打开数据库连接
```

```
                dr = cmd.ExecuteReader();                    // 将检索的记录行填充到 DataReader 对象中
                if (dr.Read())      // 如果有记录，显示该记录
                {
                    LabelTitle.Text = System.Web.HttpUtility.HtmlEncode(dr.GetString(2));
                    LabelContent.Text = System.Web.HttpUtility.HtmlEncode(dr.GetString(3));
                    LabelCreateTime.Text = dr.GetDateTime(4).ToString();
                    LabelIP.Text = dr.GetString(5);
                    LabelUserLoginName.Text = dr.GetString(1);
                }
                dr.Close();
            }
            catch (OleDbException oledbException)
            {    Response.Write(oledbException.Message);    }    // 显示连接异常信息
            finally
            {    // 如果连接打开则关闭连接
                if (conn.State == ConnectionState.Open)
                    conn.Close();
            }
        }
```

（4）发布帖子页面 TopicAdd.aspx.cs 部分代码如下：

```
        protected void ButtonOK_Click(object sender, EventArgs e)
        {    // 新建数据库连接 conn，连接到 Access 数据库
            System.Data.OleDb.OleDbConnection conn = new OleDbConnection();
                conn.ConnectionString =
                 ConfigurationManager.ConnectionStrings["MyBBSConnectionString"].ConnectionString;
                OleDbCommand cmd = new OleDbCommand();            // 新建 Command 对象
            cmd.Connection = conn;
            cmd.CommandText =
         "INSERT INTO [Topic]([UserLoginName],[ TopicTitle],[ TopicContent],[CreateTime],[IP])
                    VALUES(@UserLoginName,@TopicTitle,@TopicContent,@CreateTime,@IP)";
            cmd.CommandType = CommandType.Text;
            // 添加查询参数对象,并给参数赋值
                cmd.Parameters.AddWithValue("@UserLoginName ", Session["login_name"].ToString());
                cmd.Parameters.AddWithValue("@TopicTitle ", TextBoxTitle.Text);
                cmd.Parameters.AddWithValue("@TopicContent ", TextBoxContent.Text);
                cmd.Parameters.AddWithValue("@CreateTime ", DateTime.Now.ToString());
                cmd.Parameters.AddWithValue("@IP ", Request.ServerVariables["REMOTE_HOST"]);
            try
            {
                    conn.Open();                                    // 打开数据库连接
                    cmd.ExecuteNonQuery();                          //将添加记录
                    Response.Redirect("TopicList.aspx");
                }
            catch (OleDbException oledbException)
            {    Response.Write(oledbException.Message);    } // 显示连接异常信息
            finally
            {    // 如果连接打开则关闭连接
                if (conn.State == ConnectionState.Open)
                    conn.Close();
            }
        }
        protected void ButtonBack_Click(object sender, EventArgs e)
        {    Response.Redirect("TopicList.aspx");    }
```

（5）修改帖子页面 TopicUpdate.aspx.cs 部分代码如下：

```
protected void Page_Load(object sender, EventArgs e)
{   // 如果首次加载，调用 InitData()
    if (!IsPostBack)
        InitData();
}
protected void ButtonUpdate_Click(object sender, EventArgs e)
{    // 新建数据库连接 conn，连接到 Access 数据库
    System.Data.OleDb. OleDbConnection conn = new OleDbConnection();
    conn.ConnectionString =
      ConfigurationManager.ConnectionStrings["MyBBSConnectionString"].ConnectionString;
        OleDbCommand cmd = new OleDbCommand();                    // 新建 Command 对象
    cmd.Connection = conn;
    cmd.CommandText =
    "UPDATE [Topic] SET [TopicTitle]=@ TopicTitle,[ TopicContent]=@ TopicContent
    WHERE [TopicID]=@TopicID";
    cmd.CommandType = CommandType.Text;
    // 添加查询参数对象,并给参数赋值
        cmd.Parameters.AddWithValue("@TopicTitle ", TextBoxTitle.Text);
        cmd.Parameters.AddWithValue("@TopicContent ", TextBoxContent.Text);
   cmd.Parameters.AddWithValue("@TopicID ", Convert.ToInt32(Request.QueryString["topic_id"]);
    try
    {
            conn.Open();                                          // 打开数据库连接
            cmd.ExecuteNonQuery();                                //将添加记录
            Response.Redirect("TopicList.aspx");
    }
    catch (OleDbException oledbException)
    {    Response.Write(oledbException.Message);    }    // 显示连接异常信息
        finally
        {    // 如果连接打开则关闭连接
            if (conn.State == ConnectionState.Open)
                conn.Close();
    }
}
protected void ButtonBack_Click(object sender, EventArgs e)
{    Page.Response.Redirect("TopicList.aspx");    }
private void InitData()
{    // 获取链接传递的参数值
    int topicID = Convert.ToInt32(Request.QueryString["topic_id"]);
    // 新建数据库连接 conn，连接到 Access 数据库
        System.Data.OleDb. OleDbConnection conn = new OleDbConnection();
    conn.ConnectionString =
          ConfigurationManager.ConnectionStrings["MyBBSConnectionString"].ConnectionString;
        OleDbDataReader dr;                                       // 新建 DataReader 对象
    OleDbCommand cmd = new OleDbCommand();
    cmd.Connection = conn;
    cmd.CommandText = "SELECT * FROM [Topic] WHERE TopicID=@TopicID";
    cmd.CommandType = CommandType.Text;
    // 添加查询参数对象，并给参数赋值
      cmd.Parameters.AddWithValue("@TopicID ", topicID);
    try
```

```
        {
                conn.Open();                    // 打开数据库连接
                dr = cmd.ExecuteReader();       // 将检索的记录行填充到 DataReader 对象中
                if (dr.Read())      // 如果有记录，显示记录
                {
                    TextBoxTitle.Text = dr.GetString(2);
                    TextBoxContent.Text = dr.GetString(3);
                    LabelCreateTime.Text = dr.GetDateTime(4).ToString();
                    LabelIP.Text = dr.GetString(5);
                    LabelUserLoginName.Text = Session["login_name"].ToString();
                }
                dr.Close();
            }
            catch (OleDbException oledbException)
        {   Response.Write(oledbException.Message);   }     // 显示连接异常信息
        finally
        {   // 如果连接打开则关闭连接
                if (conn.State == ConnectionState.Open)
                    conn.Close();
        }
    }
```

8.2 实例 2——公文管理系统

本节介绍应用 ASP.NET 技术开发的一个简化公文管理系统。该系统综合运用了数据库解决方案、封装、母版页及 CSS 样式等，实现基于 Web 的公文发布、接收、浏览和查询等功能。系统由多个网站模块组成。每个模块由一组页面及相关程序组成，完成相对独立的任务，如公文浏览、发文处理和收文处理等。模块涉及与用户的交互过程，包含的文件数目和类型较多，并需要访问数据库。因此，设计好各页面内容，规划好页面之间传递的数据及对数据库的访问，对于系统的设计和实现非常重要，同时也能够为今后系统的维护和升级带来方便。

8.2.1 系统功能

开发应用系统的首要工作是进行需求分析，根据应用需求，设计系统功能。公文管理系统需要对公文进行收发文处理、收发文查阅和查询，据此系统由发文浏览、收文浏览、文件查询、发文处理、收文处理和用户登录等功能模块组成。

（1）用户登录。为了保证系统使用的安全性，进入系统首先要登录。用户成功登录系统后，系统使用 Session 变量记录已登录的用户信息，在执行各功能之前都需要进行登录检查，只有已正常登录的用户才可使用系统功能。

（2）发文浏览。列出所有发文的编号、标题和时间等信息，用户单击编号即可进入发文详情页面查看详细内容。

（3）收文浏览。其功能与发文浏览相似，列出所有收文的编号、标题和时间等信息，用户单击编号即可进入收文详情页面查看详细内容。

（4）发文处理。用于签发公文。用户填写文件标题、编号、有效期等信息，点击“发文”按钮后提交，即可发布文件。

（5）收文处理。用于签收公文。用户填写文件标题、编号、有效期等信息，点击“收文”按钮后提交，即可实现收文登记处理。

（6）文件查询。可根据文件类型（发文、收文）、标题关键字、发（收）文日期等进行文件查询。符合条件的文件信息以列表形式呈现，单击编号进入发（收）文详情页面查看详细内容。

8.2.2　数据库设计

（1）数据库结构

本系统选用 Access 数据库，所使用的数据库名为 oadata.mdb，包括 4 个数据表，分别是：

userpass——用户信息表；

wdlx——文件类型表；

wddata——文件信息表；

part——部门信息表。

各表的结构分别列于表 8-3～表 8-6 中。

表 8-3　userpass 数据表结构

字 段 名	数据类型	可否为空	说　明
Userid	文本	否	用户编号，主键
Partid	文本	否	该用户所属部门编号
Username	文本	可	用户姓名
Password	文本	否	用户密码

表 8-4　wdlx 数据表结构

字 段 名	数据类型	可否为空	说　明
Id	数值	否	自动编号(记录号)
lx	文本	否	文档类型(发文、收文)
zh	文本	否	文档字号

表 8-5　wddata 数据表结构

字 段 名	数据类型	可否为空	说　明
Id	数值	否	自动编号(作为文件编号)
zh	文本	否	文档字号
lx	文本	否	文档类型
Username	文本	否	用户姓名
Partid	文本	否	用户所属部门编号
SendDate	日期/时间	否	签发时间
ExpireDate	日期/时间	否	过期时间
Title	文本	否	文件标题
Content	文本	否	文件名

表 8-6　part 数据表结构

字 段 名	数据类型	可否为空	说　明
Id	数值	否	自动编号(记录号)
Partid	文本	否	部门编号
Partname	文本	否	部门名称

（2）配置数据库连接

建立数据库文件 oadata.mdb 后，将其保存在所创建的网站项目的“App_Data”文件夹下，在项目中配置 web.config 中的数据库连接参数如下：

```
    <connectionStrings>
```

```
    <add name="documentsConnectionString" connectionString="Provider=Microsoft.Jet.OLEDB.4.0;
    Data Source=|DataDirectory|oadata.mdb" providerName="System.Data.OleDb"/>
  </connectionStrings>
```

（3）数据库访问类

由于多个功能模块都要执行数据库操作，因此将数据库操作功能设计为 DBHelper 类，将其保存在网站项目的“ App_Code ”文件夹下，DBHelper 类内容如下：

```
public class DBHelper
{
    protected OleDbConnection Connection;          //存放连接对象
    protected string ConnectionString;             //存放连接串
    public DBHelper() //构造函数
    {
        ConnectionString =
            WebConfigurationManager.ConnectionStrings["documentsConnectionString"].
              ConnectionString;
    }
    private void Open()   //公有方法，建立数据库连接
    {   //判断数据库连接是否存在
        if (Connection == null)
        {    //不存在，新建并打开
            Connection = new OleDbConnection();
            Connection.ConnectionString = ConnectionString;
            Connection.Open();
        }
        else
        {   //存在，判断是否处于关闭状态
            if (Connection.State.Equals(ConnectionState.Closed))
                Connection.Open();      //连接处于关闭状态，重新打开
        }
    }
    public void Close()    //公有方法，关闭数据库连接
    {    if (Connection.State.Equals(ConnectionState.Open))
        {       Connection.Close();       //连接处于打开状态，关闭连接
        }
    }
    public   DataSet GetDataSetSql(string SqlString,string tableName)
    {//公有方法，根据 Sql 语句，返回一个结果数据集 DataSet
        Open();
        if (Connection != null)
        {    OleDbDataAdapter Adapter = new OleDbDataAdapter(SqlString, Connection);
            DataSet Ds = new DataSet();
            Adapter.Fill(Ds, tableName);
            Close();
            return Ds;
        }
        else   return null;
    }
    public   int ExecuteSql(string SqlStr)
    { //公有方法，根据 Sql 语句，执行并返回影响结果的行数
        int Count = -1;
        Open();
        OleDbCommand cmd = new OleDbCommand(SqlStr, Connection);
```

```
            Count = cmd.ExecuteNonQuery();
            Close();
            return Count;
        }
    }
```

8.2.3　各子系统设计与程序代码

（1）创建母版页

母版页提供了类似模板的机制。本系统将 logo、功能导航栏和页脚等各页面的公共内容设计为母版页，在母版页中定义内容区域，各功能模块只需将内容页放置到内容区域即可。母版页面如图 8-6 所示。

图 8-6　公文管理系统母版页

母版页主要代码如下：

```
<%@ Master Language="C#" AutoEventWireup="true" CodeFile="MasterPage.master.cs"
    Inherits="MasterPage" %>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" >
<head runat="server"><title>公文管理</title>
    <link href="StyleSheet.css" rel="stylesheet" type="text/css" /></head>
<body> <form id="form1" runat="server">
  <div id="maindiv"> <div id="HeadDiv"> <br /><br /></div>
  <div id="MenuDiv">｜ 
  <asp:HyperLink ID="HyperLink2" runat="server" NavigateUrl="~/login.aspx">首页</asp:HyperLink>
  <asp:HyperLink ID="HyperLink3" runat="server" NavigateUrl="~/Send.aspx">发文浏览</asp:HyperLink>
  <asp:HyperLink ID="HyperLink4" runat="server" NavigateUrl="~/Receive.aspx?id=%">
    收文浏览</asp:HyperLink>
<asp:HyperLink ID="HyperLink5" runat="server" NavigateUrl="~/Query.aspx?id=%">
    文件查询</asp:HyperLink>
<asp:HyperLink ID="HyperLink6" runat="server" NavigateUrl="~/SendProc.aspx">
    发文处理</asp:HyperLink>
  <asp:HyperLink ID="HyperLink7" runat="server" NavigateUrl="~/ReceiveProc.aspx">
    收文处理</asp:HyperLink></div>
<div id="ContentDiv" style="background-image: url(images/Dlhbback.gif);">
<asp:ContentPlaceHolder ID="ContentPlaceHolder1" runat="server"></asp:ContentPlaceHolder>
    <br /></div>
<div id="EndDiv">CopyRight; 2014-2015 Websoft All Right Reserved.版权所有 E-mail:webmaster@test.com
</div></div></form></body></html>
```

母版页采用 DIV+CSS 方式设计，将页面设计为如下几个块：① maindiv：全局页；② HeadDiv：页首；③ MenuDiv：功能导航栏；④ ContentDiv：内容区域；⑤ EndDiv：页脚。各块的样式分

别进行定义。

（2）登录页面

该页面提供用户登录界面，如图 8-7 所示。当用户成功登录后，用 Session 变量记录其用户信息，包括用户名、所在单位等。

图 8-7　登录页面

登录页面采用 table 控件组织信息，文件名为 Login.aspx，内容如下：

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
        CodeFile="login.aspx.cs" Inherits="login" Title="公文信息管理系统" %>
 <asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server">
   <table align="center" style="text-align: center" border="1">
    <tr><td bgcolor="buttonshadow" colspan="3" style="height: 16px; text-align: center">
        <span style="font-size: 9pt">用户登录</span></td> </tr>
    <tr> <td style="width: 100px; height: 26px;" colspan="">
        <asp:Label ID="Label1" runat="server" Text="用户名" Font-Size="10pt"></asp:Label></td>
        <td colspan="2" style="height: 26px">
        <asp:TextBox ID="txtid" runat="server" MaxLength="16" Width="120px"></asp:TextBox>
        <asp:RequiredFieldValidator ID="RequiredFieldValidator1" runat="server"
ControlToValidate="txtid"   ErrorMessage="账号不能为空" Font-Size="10pt">
</asp:RequiredFieldValidator></td></tr>
    <tr><td style="width: 100px; height: 26px" colspan="">
         <asp:Label ID="Label2" runat="server" Text="密码" Font-Size="10pt"></asp:Label></td>
         <td colspan="2" style="height: 26px">
         <asp:TextBox ID="txtpassword" runat="server" TextMode="Password" MaxLength="16"

Width="120px"></asp:TextBox>
         <asp:RequiredFieldValidator ID="RequiredFieldValidator2" runat="server"
         ControlToValidate="txtpassword"ErrorMessage="密码不能为空" Font-Size="10pt">
         </asp:RequiredFieldValidator></td></tr>
   <tr><td style="width: 100px; height: 27px;" colspan=""></td>
        <td style="height: 27px;" colspan="2">
        <asp:Button ID="Button1" runat="server" Text="确定" OnClick="BtnOK_Click" />
         <asp:Button ID="Button2" runat="server" Text="取消" OnClick="BtnDel_Click" /></td></tr>
   <tr><td colspan="3">
         <asp:Label ID="Label3" runat="server" Font-Size="9pt" ForeColor="Red"></asp:Label></td></tr>
</table><br /></asp:Content>
```

其中，第一行声明为引用母版页，MasterPage.master 为文件名。两个按钮控件的事件处理程序如下：

```
protected void BtnOK_Click(object sender, EventArgs e)
{   string strId = txtid.Text;
    string strPass = txtpassword.Text;
```

```
            DBHelper dh = new DBHelper();   //创建数据操作类实例
            string strSql = "SELECT * FROM userpass WHERE Userid='" + strId + "' AND Passwd='" + strPass
+ "'";
            DataSet DS = dh.GetDataSetSql(strSql,"userpass");
            if (DS.Tables[0].Rows.Count==0)
            {   Label3 .Text ="账号或密码有错，请重新输入";      }
            else //登录成功，用 Session 变量保存用户信息
            {   Session["Userid"]=txtid .Text ;
                Session["Username"] = DS.Tables[0].Rows[0]["Username"].ToString();
                Session["Userpart"] = DS.Tables[0].Rows[0]["Partid"].ToString();
                Response.Redirect("LoginSucc.aspx");      //转向登录成功页
            }
        }
    protected void BtnDel_Click(object sender, EventArgs e)
    {   txtid.Text = "";   txtpassword.Text = "";   }
```

LoginSuccess.aspx 是一个简单的网页，仅显示一行文字和一幅图像，如图 8-8 所示，此时用户就可使用系统的功能了。

图 8-8　登录成功页面

（3）发文浏览和发文详情

发文浏览界面如图 8-9 所示。首先列出发文的编号、文件标题和发文日期，其中编号为超链接按钮，当用户单击某个发文编号后，将进入发文详情页面，显示该发文的详情，如图 8-10 所示。两个页面的文件名分别为 Send.aspx 和 showDoc.aspx。

图 8-9　发文浏览页面

图 8-10　发文详情页面

发文浏览界面中采用 GridView 控件，显示文档摘要信息（ID、文号、标题和日期）。发文详情界面中 DataList 控件显示文档详细内容。

Send.aspx 页面部分内容如下：

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
    CodeFile="Send.aspx.cs" Inherits="Send" Title="公文信息管理系统" %>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server"><br />
```

```
<div id="content1" align="center">
  <asp:GridView ID="sendDocumnet" runat="server" AutoGenerateColumns="False"
      DataKeyNames="ID" DataSourceID="SqlDataSource1" Width="700px" AllowPaging="True">
<Columns>
  <asp:BoundField DataField="ID" HeaderText="ID" InsertVisible="False"
  ReadOnly="True" SortExpression="ID" Visible="False" />
  <asp:HyperLinkField DataNavigateUrlFields="ID" DataNavigateUrlFormatString="showDoc.aspx?id={0}"
  DataTextField="Id" HeaderText="ID" />
    <asp:BoundField DataField="zh" HeaderText="文号" SortExpression="zh" />
    <asp:BoundField DataField="Title" HeaderText="标题" SortExpression="Title" />
    <asp:BoundField DataField="SendDate" HeaderText="发文日期" SortExpression="SendDate" />
  </Columns></asp:GridView></div>
</asp:Content>
```

showDoc.aspx 页面部分的内容如下：

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
    CodeFile="showDoc.aspx.cs" Inherits="showDoc" Title="公文信息管理系统" %>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server"><br />
  <div align="center"> <asp:DataList ID="DataList1" runat="server" Height="121px"
      RepeatColumns="1" RepeatDirection="Horizontal"    Width="524px">
<ItemTemplate>
  <table border="0" cellpadding="0" cellspacing="0" style="width: 100%; height: 100%">
  <tr><td style="width: 100%; height: 20%; font-size: 20pt; text-align: center;"> <br />
      <%# DataBinder.Eval(Container.DataItem, "Title")%><br /><br /></td></tr>
  <tr><td style="width: 100%; height: 10%">
      文号：【<%# DataBinder.Eval(Container.DataItem, "zh")%>】  
      发文时间：<%# DataBinder.Eval(Container.DataItem, "SendDate")%>
【<a href='javascript:window.print()'>打印此页</a>】【<a href='javascript:history.back()'>返回</a>】
      </td></tr>
    <tr><td style="width: 100%; height: 70%; text-align: justify; line-height: 17pt; ">
      <br />  
      <%# FormatBody(DataBinder.Eval(Container.DataItem,"Content").ToString()) %><br /></td></tr>
    </table></ItemTemplate></asp:DataList><br /></div>
</asp:Content>
```

进入 Send.aspx 页面时通过 Page_Load()事件处理程序，判断是否是登录用户。程序如下：

```
protected void Page_Load(object sender, EventArgs e)
{   if (Session["Userid"] == null )
    {   Response.Write("<script>alert('登录错误');window.location.href='./login.aspx'</script>");   }
}
```

单击发文列表中的 Id，进入 showDoc.aspx 页面，该页面的 Page_Load()事件处理程序，通过传入的 Id 号检索 wddata 表，显示发文详情。程序如下：

```
protected void Page_Load(object sender, EventArgs e)
{    DBHelper dh = new DBHelper();
     string strSql = "select * from wddata where Id=" + Request.Params["id"].ToString();
     DataSet DS = dh.GetDataSetSql(strSql, "wddata");
     DataList1.DataSource = DS.Tables[0].DefaultView;
     DataList1.DataBind();
 }
```

收文浏览与发文浏览十分相似，限于篇幅，这里不再给出。

（4）文件查询

文件查询界面如图 8-11 所示。选择“公文类型”和“收（发）日期”，也可输入“标题关键

字”，单击“查询”按钮，即可将符合条件的文档列于表中，如图 8-12 所示。

图 8-11　文件查询界面

图 8-12　文件查询结果

该设计主要使用日历控件和数据库查询操作。程序名为 Query.aspx，页面内容如下：

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
        CodeFile="Query.aspx.cs" Inherits="Query" Title="公文信息管理系统" %>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server"><br />
<div id="content1" align="left"><asp:Label ID="Label1" runat="server" Text="公文类型">
      </asp:Label>
   <asp:DropDownList ID="DplLx" runat="server">
         <asp:ListItem>发文</asp:ListItem>
         <asp:ListItem>收文</asp:ListItem>
   </asp:DropDownList>
   <asp:Label ID="Label2" runat="server" Text="标题关键字"></asp:Label> <asp:TextBox
         ID="TxtTitle" runat="server" Width="148px"></asp:TextBox>  
   <asp:Button ID="BtnQuery" runat="server" Text="查询" onclick="BtnQuery_Click" />  收（发）日期
   <table align="center" style="text-align: center" border="0">
```

```
        <tr><td><asp:Calendar ID="Calendar1"
              runat="server" BackColor="#FFFFCC" BorderColor="#FFCC66" BorderWidth="1px"
              DayNameFormat="Shortest" Font-Names="Verdana" Font-Size="8pt"
              ForeColor="#663399" Height="150px" ShowGridLines="True" Width="196px">
              <DayHeaderStyle BackColor="#FFCC66" Font-Bold="True" Height="1px" />
              <NextPrevStyle Font-Size="9pt" ForeColor="#FFFFCC" />
              <OtherMonthDayStyle ForeColor="#CC9966" />
              <SelectedDayStyle BackColor="#CCCCFF" Font-Bold="True" />
              <SelectorStyle BackColor="#FFCC66" />
              <TitleStyle BackColor="#990000" Font-Bold="True" Font-Size="9pt" ForeColor="#FFFFCC" />
              <TodayDayStyle BackColor="#FFCC66" ForeColor="White" />
              </asp:Calendar></td>
            <td> <asp:Label ID="Label3" runat="server" Text="到"></asp:Label></td>
            <td><asp:Calendar ID="Calendar2"
              runat="server" BackColor="#FFFFCC" BorderColor="#FFCC66" BorderWidth="1px"
              DayNameFormat="Shortest" Font-Names="Verdana" Font-Size="8pt"
              ForeColor="#663399" Height="151px" ShowGridLines="True" Width="202px">
              <DayHeaderStyle BackColor="#FFCC66" Font-Bold="True" Height="1px" />
              <NextPrevStyle Font-Size="9pt" ForeColor="#FFFFCC" />
              <OtherMonthDayStyle ForeColor="#CC9966" />
              <SelectedDayStyle BackColor="#CCCCFF" Font-Bold="True" />
              <SelectorStyle BackColor="#FFCC66" />
              <TitleStyle BackColor="#990000" Font-Bold="True" Font-Size="9pt" ForeColor="#FFFFCC" />
              <TodayDayStyle BackColor="#FFCC66" ForeColor="White" />
          </asp:Calendar></td></tr></table><br />
          <div id="content2" align="center">
          <asp:GridView ID="sendDocumnet" runat="server" AutoGenerateColumns="False"
              DataKeyNames="ID" Width="700px" AllowPaging="True">
           <Columns>
              <asp:BoundField DataField="ID" HeaderText="ID" InsertVisible="False"
                    ReadOnly="True" SortExpression="ID" Visible="False" />
              <asp:HyperLinkField DataNavigateUrlFields="ID"
                    DataNavigateUrlFormatString="showDoc.aspx?id={0}" DataTextField="Id"
                    HeaderText="ID" />
              <asp:BoundField DataField="zh" HeaderText="文号" SortExpression="zh" />
              <asp:BoundField DataField="Title" HeaderText="标题" SortExpression="Title" />
             <asp:BoundField DataField="SendDate" HeaderText="发文日期" SortExpression="SendDate" />
            </Columns></asp:GridView><br /></div><br /></div>
    </asp:Content>
```

单击“查询”按钮，触发 BtnQuery_Click()事件处理，从 wddata 表中查询符合条件的记录，绑定到 GridView 控件上。程序如下：

```
    protected void BtnQuery_Click(object sender, EventArgs e)
    {   string lx = DplLx.Text;
        string title=TxtTitle.Text;
        string date1 = Calendar1.SelectedDate.ToString();
        string date2 = Calendar2.SelectedDate.ToString();
        string strSql = "select * from wddata where lx='" + lx +"' and SendDate >= #" + date1 + "# and
            SendDate<= #" + date2 + "#";
        if (title != "")
            strSql = strSql +" and Title like '%" + title + "%'";
        DBHelper dh = new DBHelper(); //创建数据操作类实例
        DataSet DS = dh.GetDataSetSql(strSql, "wddata");
```

```
        sendDocumnet.DataSource = DS.Tables[0].DefaultView;
        sendDocumnet.DataBind();
    }
```

（5）发文处理

发文处理界面如图 8-13 所示。用户填写文件标题，选择文件编号、字号、有效日期，填写发文内容，单击“发文”按钮，即可完成发文操作。这一部分主要是使用 SQL 的 insert 语句对数据表进行插入操作。

图 8-13　发文处理界面

程序名为 SendProc.aspx，页面内容如下：

```
<%@ Page Language="C#" MasterPageFile="~/MasterPage.master" AutoEventWireup="true"
    CodeFile="SendProc.aspx.cs" Inherits="SendProc" Title="公文信息管理系统" %>
<asp:Content ID="Content2" ContentPlaceHolderID="ContentPlaceHolder1" Runat="Server"><br />
<div id="content1" align="left">
    <asp:Label ID="Label1" runat="server" Text="文件标题"></asp:Label> 
    <asp:TextBox ID="TxtTitle" runat="server" Width="377px"></asp:TextBox><br /><br />
    <asp:Label ID="Label2" runat="server" Text="文件编号"></asp:Label> 
    <asp:DropDownList ID="DplZh" runat="server">
        <asp:ListItem>校发</asp:ListItem>
        <asp:ListItem>财发</asp:ListItem>
        <asp:ListItem>教发</asp:ListItem> </asp:DropDownList>
        <asp:Label ID="Label3" runat="server" Text="字第"></asp:Label> 
    <asp:TextBox ID="TxtNo" runat="server" Width="37px"></asp:TextBox> 
    <asp:Label ID="Label4" runat="server" Text="号"></asp:Label>
    <asp:Label ID="Label5" runat="server" Text="有效日期"></asp:Label> 
    <asp:DropDownList ID="DplYear" runat="server">
        <asp:ListItem Value="2014">2014 年</asp:ListItem>
        <asp:ListItem Value="2015" selected="True">2015 年</asp:ListItem>
        <asp:ListItem Value="2016">2016 年</asp:ListItem></asp:DropDownList> 
    <asp:DropDownList ID="DplMonth" runat="server">
        <asp:ListItem Value="1">1 月</asp:ListItem>
        <asp:ListItem>2 月</asp:ListItem>
        <asp:ListItem Value="3">3 月</asp:ListItem>
        <asp:ListItem Value="4">4 月</asp:ListItem>
        <asp:ListItem Value="5">5 月</asp:ListItem>
        <asp:ListItem Value="6">6 月</asp:ListItem>
```

```
                <asp:ListItem Value="7">7 月</asp:ListItem>
                <asp:ListItem Value="8">8 月</asp:ListItem>
                <asp:ListItem Value="9">9 月</asp:ListItem>
                <asp:ListItem Value="10">10 月</asp:ListItem>
                <asp:ListItem Value="11">11 月</asp:ListItem>
                <asp:ListItem Value="12">12 月</asp:ListItem>
            </asp:DropDownList>
            <asp:TextBox ID="TxtDate" runat="server" Width="45px"></asp:TextBox> 
            <asp:Label ID="Label6" runat="server" Text="日"></asp:Label><br /><br />
            <asp:Label ID="Label7" runat="server" Text="发文内容"></asp:Label> <br />
            <asp:TextBox ID="TxtContent" runat="server" Height="267px" TextMode="MultiLine"
                Width="587px"></asp:TextBox><br />
              <div align="center"><asp:Button ID="BtnSend" runat="server" Text="发文"
                    onclick="BtnSend_Click" /></div><br /><br /></div>
        </asp:Content>
```

单击“发文”按钮，触发 BtnSend_Click()事件处理，向 wddata 表中插入记录。程序如下：

```
protected void BtnSend_Click(object sender, EventArgs e)
{    string title = TxtTitle.Text;
     string lx="发文";
     string zh = DplZh.Text + "第" + TxtNo.Text + "号";
     string date2 = DplYear.SelectedValue + "/" + DplMonth.SelectedValue + "/" + TxtDate.Text;
     string date1 = System.DateTime.Now.Date.ToString();
     string content = TxtContent.Text;
     string strSql;
     strSql = "insert into wddata(Title,lx,zh,Username,Partid,SendDate,ExpireDate,Content) values
         ('" + title + "','" + lx + "','" + zh + "','" + Session["Username"] + "','";
     strSql = strSql + Session["Userpart"] + "',#" + date1 + "#,#" + date2 + "#,'" + content + "')";
     DBHelper dh = new DBHelper();
     dh.ExecuteSql(strSql);
     Response.Write("<script>alert(\"发文成功！\");</script>");
     TxtTitle.Text = "";
     TxtNo.Text = "";
     TxtDate.Text = "";
     TxtContent.Text = "";
}
```

收文处理与发文处理基本相同，请读者参照发文处理自行设计。

本 章 小 结

本节分别通过“BBS 论坛管理系统”和“公文管理系统”实例，介绍了 ASP.NET 应用程序的开发过程和要点。要设计好一个 ASP.NET 应用系统，需要对应用需求进行详细分析，在此基础上进行功能设计和数据库设计。在系统实现上，要综合运用多种服务器控件和对象，特别是数据库访问技术。因此，熟练掌握并灵活运用各种服务器控件、对象和数据库操作，是保证顺利完成系统开发的关键。

附录　HTML、JavaScript、CSS、ASP.NET 实用列表

附录 A　HTML 语言常用标记和属性

1．文件头标记

<head>，</head>：HTML 文件头部开始和结束标记。

2．文件标题标记

<title>，</title>：HTML 文件标题，是显示于浏览器标题栏的字符串。

3．样式标记

<style>，</style>：CSS 样式定义标记。属性 type，指明样式的类别，默认值为 text/css。

4．搜索引擎标记

<meta>：为搜索引擎定义页面主题及页面刷新等信息。包括如下属性。

- name　　meta 名字。
- http-equiv　　说明 content 属性内容的类别。
- content　　定义页面内容，一些特定内容要与 http-equiv 属性配合使用。

例如：http-equiv="refresh"，则 content 中是页面刷新的时间；http-equiv="content-language"，则 content 中是页面语言；http-equiv="PICS-Label"，则 content 中是页面内容的等级；http-equiv="expires"，则 content 中是页面过期的日期。

5．文件体标记

<body>，</body>：表明 HTML 文件体部的开始和结束，body 标记的属性列于表 A-1 中。

表 A-1　body 标记属性

属　性	取　值	含　义	默 认 值
bgcolor	颜色值	页面背景颜色	#FFFFFF
text	颜色值	HTML 文件中文字的颜色	#000000
link	颜色值	HTML 文件中待链接的超链接对象的颜色	
alink	颜色值	HTML 文件中链接中的超链接对象的颜色	
vlink	颜色值	HTML 文件中已链接的超链接对象的颜色	
background	图像文件名	页面的背景图像	无
topmargin	整数	页面显示区距窗口上边框的距离，以像素点为单位	0
leftmargin	整数	页面显示区距窗口左边框的距离，以像素点为单位	0

6．图像标记

<img>：向页面中插入一幅图像。<img>标记的属性有如下几种。

- src 指定图像文件的地址，该属性值必须指明。值的形式可以是一个本地文件名，也可以是一个 URL。
- border 指定图像边框的粗细，值为整数。若为 0，表示无边框；值越大，边框越粗。
- width 指定图像宽度，值为整数，单位为屏幕像素点。若不指出该属性值，则浏览器根据图像的实际尺寸显示。
- height 指定图像高度，值为整数，单位为屏幕像素点。若不指出该属性值，则浏览器根据图像的实际尺寸显示。
- alt 若设置了该属性值，则当鼠标移至该图像区域时，将以一个小标签显示该属性的值。

7．文字显示和段落控制标记

文字显示属性主要有字体、字号、颜色，段落控制显示对象的分段。常用的文字显示和段落控制标记列于表 A-2 中。

表 A-2　文字显示和段落控制标记

标　记	含　义
,	分别以属性 face, size, color 控制字体、字号、字的颜色显示特性
<I>,</I>	斜体显示
<B>,</B>	粗体显示
<U>,</U>	加下画线显示
_,	下标字体
[,]	上标字体
<big></big>	大字体
,	小字体
<h1>～<h6>	标题，数字越大，显示的标题字越小
,	分段标记，属性有布局方式 align：left—左对齐；center—居中对齐；right—右对齐
,	块容器标记，其中的内容是一个独立段落
<hr>	分隔线，属性有：width（线的宽度）、color（线的颜色）
,	居中显示

8．超链接标记

<a>，：创建超链接。它有以下两种属性。

- href 指出目标页面的 URL。
- target 指明目标页面显示的窗口。

9．列表标记

，，
，，<dl>，</dl>分别为无序列表、有序列表和定义列表，定义内部需使用<li>给出各表项。

10．预定格式标记

<pre>，</pre>：预定格式的信息。

11．表格标记

<table>，</table>：定义表格的开始和结束。其属性如表 A-3 所示。

表 A-3　table 标记属性

属　性	取　值	含　义	默 认 值
border	整数	表格边框粗细，该值为 0，则表格没有边框；值越大，则表格边框越粗	0
width	百分比值	表格宽度，以相对于充满窗口的百分比计，如 60%	100%
	整数	表格宽度，以屏幕像素点计	
cellpadding	整数	每个表项内容与表格边框之间的距离，以像素点为单位	0
cellspacing	整数	表格边框之间的距离，以像素点为单位	2
bordercolor	颜色值	表格边框的颜色	#000000
background	图像文件名	表格的背景图像	无
align	left|center|right	表格的位置	left

<tr>，</tr>：定义表格一行的开始和结束。其属性如表 A-4 所示。

表 A-4　tr 标记属性

属　性	取　值	含　义	默 认 值
align	left|center|right	本行各表格项的横向排列方式	left（左对齐）
bgcolor	颜色值	本行各表格项的背景色	#000000
valign	top|middle|bottom	本行各表格项的纵向排列方式	middle
width	百分比值|整数	本行宽度（受<table>的 width 属性值制约）	
height	整数	本行高度，以像素点为单位	

<td>，</td>：定义表格中一个单元格的开始和结束。其属性如表 A-5 所示。

表 A-5　td 标记属性

属　性	取　值	含　义	默 认 值
align	left|center|right	本表格项的横向排列方式	left（左对齐）
bgcolor	颜色值	本表格项的背景色	#000000
valign	top|middle|bottom	本表格项的纵向排列方式	middle
width	百分比值|整数	本表格项宽度	
height	整数	本表格项高度，以像素点为单位	
background	图像文件名	本表格项的背景图像	无
colspan	整数	按列横向结合	1
rowspan	整数	按行纵向结合	1

12．表单标记

<form>，</form>：定义表单的开始和结束。包括以下属性。

- method（方法）属性：取值为 post 或 get。
- action 属性：指出用户所提交的数据将由哪个服务器的哪个程序处理，可处理用户提交的数据的服务器程序种类较多，如 CGI 程序、ASP 脚本程序、PHP 程序等。

<textarea>：允许输入多行文字。

<select>：下拉列表选择。以<option>标记给出各选项。

<input>：定义表单的输入域，由 type 属性值给出。可由 type 属性给出的输入域类型如表 A-6 所示。

表 A-6　表单 input 标记定义的输入域

输　入　域	说　　明
Text（文本框）	输入一行文字
Radio（单选钮）	当有多个选项时，只能选其中一项
Checkbox（复选框）	当有多个选项时，可以选其中多项
Submit（提交按钮）	将数据传递给服务器
Password（密码输入框）	用户输入的字符以“*”显示
Reset（重置按钮）	将用户输入的数据清除
Hidden（隐藏域）	在浏览器中不显示，但可通过程序取其值或改变值。主要用于浏览器向服务器传递数据而不想让浏览器用户知道的情形
Button（按钮）	普通按钮，按下后的操作需设计程序完成

<input>标记的其他属性还有以下两种。

- name　　输入域的名称。
- value　　输入域的值。

13．框架标记

<frameset>，</frameset>：定义框架特性。其属性如表 A-7 所示。

表 A-7　框架 frameset 标记属性

属　　性	取　　值	含　　义	默 认 值
rows	百分比值	将窗口上下（横向）分割，每个框架高度占整个窗口高度的百分比	无
	整数	将窗口上下（横向）分割，每个框架高度的像素点数	
cols	百分比值	将窗口左右（纵向）分割，值的格式和含义与“rows”属性类似	无
	整数		
frameborder	yes|no	帧框架边框是否显示	yes
bordercolor	颜色值	框架边框颜色	gray（灰）

<frame>：指明框架所对应的 HTML 或脚本文件。其属性如表 A-8 所示。

表 A-8　框架 frame 属性

属　　性	取　　值	含　　义	默 认 值
src	HTML 文件名	框架对应的 HTML 文件	无
name	字符串	框架的名字，可在程序和<a>标记的 target 属性中引用	无
noresize	无	不允许用户改变框架窗口大小	无
scrolling	yes|no|auto	框架边框是否出现滚动条	auto
marginwidth	整数	框架左右边缘像素点数	0
marginheight	整数	框架上下边缘像素点数	0

附录 B　JavaScript 常用对象的属性、方法、事件处理和函数

1．对象

（1）Array 对象。功能：创建并操作数组。其属性和方法见表 B-1。

表 B-1　Array 对象

属　性	说　明
length	数组中元素数
prototype	为 Array 对象添加一个属性
方　法	**说　明**
Join()	返回由数组中所有元素连接而成的字符串
reverse()	逆转数组中各元素
sort()	对数组元素排序

（2）String 对象。功能：处理字符串。其属性和方法见表 B-2。

表 B-2　String 对象

属　性	说　明
length	字符串长度
方　法	**说　明**
charAt(position)	返回 String 对象实例中位于 position 位置上的字符，其中 position 为正整数或 0。注意字符串中字符位置从 0 开始计
indexOf(str) indexOf(str,start-position)	字符串查找，str 是待查找的字符串。在 String 对象实例中查找 str，若给出 start-position，则从 start-position 位置开始查找，否则从 0 开始查找；若找到，返回 str 在 String 对象实例中的起始位置，否则返回 –1
lastIndexOf(str)	该方法与 indexOf()类似，差别在于它是从右往左查找
substring(position) substring(position1,position2)	返回 String 对象的子串。如果只给出 position，返回从 position 开始至字符串结束的子串；如果给出 position1 和 position2，则返回从二者中较小值开始的位置至较大值结束处的子串
toLowerCase() toUpperCase()	分别将 String 对象实例中的所有字符改变为小写、大写
big()	大字体显示
Italics()	斜体字显示
bold()	粗体字显示
blink()	字符闪烁显示
small()	字符小字体显示
fixed()	固定高亮字显示
fontsize(size)	控制字体大小等
Anchor()	返回一个字符串，该字符串是网页中的一个锚点名
link()	返回一个字符串，该字符串用来在网页中构造一个超链接
fontcolor(color)	返回一个字符串，此字符串可改变网页中的文字颜色
fontsize()	返回一个字符串，此字符串可改变网页中的文字大小

（3）Math 对象。功能：关于数学常量、函数。其属性和方法见表 B-3。

表 B-3　Math 对象

属　性	说　明
E	常数 e，自然对数的底，近似值为 2.718
LN2	2 的自然对数，近似值为 0.693
LN10	10 的自然对数，近似值为 2.302
LOG2E	以 2 为底，E 的对数，即 $\log_2 e$，近似值为 1.442
LOG10E	以 10 为底，E 的对数，即 $\log_{10} e$，近似值为 0.434
PI	圆周率，近似值为 3.142
SQRT1_2	0.5 的平方根，近似值为 0.707
SQRT2	2 的平方根，近似值为 1.414

续表

方　法	说　明
sin(val)	返回 val 的正弦值，val 的单位是 rad（弧度）
cos(val)	返回 val 的余弦值，val 的单位是 rad（弧度）
tan(val)	返回 val 的正切值，val 的单位是 rad（弧度）
asin(val)	返回 val 的反正弦值，val 的单位是 rad（弧度）
exp(val)	返回 E 的 val 次方
log(val)	返回 val 的自然对数
pow(bv,ev)	返回 bv 的 ev 次方
sqrt(val)	返回 val 的平方根
abs(val)	返回 val 的绝对值
ceil(val)	返回大于或等于 val 的最小整数值
floor(val)	返回小于或等于 val 的最小整数值
round(val)	返回 val 四舍五入得到的整数值
random()	返回 0～1 之间的随机数
max(val1,val2)	返回 val1 和 val2 之间的大者
min(val1,val2)	返回 val1 和 val2 之间的小者

（4）Number 对象。功能：给出了系统最大值、最小值以及非数字常量的定义。其属性见表 B-4。

表 B-4　Number 对象

属　性	说　明
MAX_VALUE	数值型最大值，值为 1.7976931348623517e+308
MIN_VALUE	数值型最小值，值为 5e–324
NaN	非合法数字值
POSITIVE_INFINITY	正无穷大
NEGATIVE_INFINITY	负无穷大

（5）Date 对象。功能：Date 对象封装了有关日期和时间的操作。属性：无。其方法见表 B-5。

表 B-5　Date 对象

方　法	说　明
getYear()	返回对象实例的年份值。如果年份在 1900 年后，则返回后两位，例如 1998 将返回 98；如果年份在 100～1900 之间，则返回完全值
getMonth()	返回对象实例的月份值，其值在 0～11 之间
getDate()	返回对象实例日期中的天，其值在 1～31 之间
getDay()	返回对象实例日期是星期几，其值在 0～6 之间，0 代表星期日
getHours()	返回对象实例时间的小时值，其值在 0～23 之间
getMinutes()	返回对象实例时间的分钟值，其值在 0～59 之间
getSeconds()	返回对象实例时间的秒值，其值在 0～59 之间
getTime()	返回一个整数值，该值等于从 1970 年 1 月 1 日 00:00:00 到该对象实例存储的时间所经过的毫秒数
getTimezoneOffset()	返回当地时区与 GMT 标准时的差别，单位是 min（GMT 时间是基于格林尼治时间的标准时间，也称 UTC 时间）
SetDate()	设置日期时间值
SetHours()	设置时间的时数
SetMinutes()	设置时间的分数

续表

方　法	说　明
SetSeconds()	设置时间的秒数
SetTime()	以整数值设置小时值，该值等于从 1970 年 1 月 1 日 00:00:00 到该对象实例存储的时间所经过的毫秒数
SetYear()	设置年份值
ToGMTString()	将日期时间值转换为 GMT 值串
ToLocalString()	将日期时间值转换为本地时间值串
ToString()	将日期时间值转换为字符串
UTC()	静态方法，将字符串参数表示的日期转换为一个整数值，该值等于从 1970 年 1 月 1 日 00:00:00 计算起的毫秒数
Parse()	静态方法，将数值参数表示的日期转换为一个整数值，该值等于从 1970 年 1 月 1 日 00:00:00 计算起的毫秒数

（6）Navigator 对象。功能：Navigator 对象包含正在使用的浏览器版本信息。其属性和方法见表 B-6。

表 B-6　Navigator 对象

属　性	说　明
appName	以字符串形式表示的浏览器名称
appVersion	以字符串形式表示的浏览器版本信息，包括浏览器的版本号、操作系统名称等
appCodeName	以字符串形式表示的浏览器代码名字，通常值为 Mozilla
userAgent	以字符串表示的完整的浏览器版本信息，包括 appName, appVersion 和 appCodeName 信息
mimeType	在浏览器中可以使用的 mime 类型
plugins	在浏览器中可以使用的插件
方　法	**说　明**
javaEnabled()	返回逻辑值，表示客户浏览器可否使用 Java

事件处理：无。

（7）Window 对象。功能：Window 对象描述浏览器窗口特征，是 Document、Location 和 History 对象的父对象。其属性、事件和方法见表 B-7。

表 B-7　Window 对象

属　性	说　明
parent	代表当前窗口或框架（frame）的父窗口
self	代表当前窗口
top	代表主窗口
window	代表当前窗口或框架（frame）的父窗口
status	浏览器当前状态栏显示的内容
defaultStatus	浏览器状态栏显示的默认值
opener	是一个窗口名，该窗口是由方法 open()打开的最新窗口
frames	是一个数组，数组的各成员是窗口内的各个框架
事　件	**说　明**
Load()	HTML 文件载入浏览器时触发，事件处理名为 onLoad
UnLoad()	离开页面时触发，事件处理名为 onUnLoad

续表

方　法	说　明
alert()	产生警告对话框
confirm()	产生带有确定和否认的对话框
prompt()	产生带有提示信息的对话框
open()	生成一个新窗口
close()	关闭一个窗口
focus()	使窗口获得焦点
blur()	使窗口失去焦点
setTimeout()	设置超时
clearTimeout()	清除超时设置
scroll()	使窗口滚动到指定位置处

（8）Document 对象。功能：一个 Document 对象对应 HTML 文件的页面。其属性和方法见表 B-8。

表 B-8　Document 对象

属　性	说　明
alinkColor	被激活的超链接文本颜色，即鼠标单击超链接时超链接文本的颜色
bgColor	页面背景颜色
fgColor	页面前景色，即页面文字的颜色
laseModified	HTML 文件最后被修改的日期，是只读属性
linkColor	未被访问的超链接的文本颜色
referrer	用户先前访问的 URL
title	HTML 文件的标题，对应<title>标记
URL	本 HTML 文件的完整的 URL
vlinkColor	已被访问过的超链接的文本颜色
anchors 数组	HTML 文件中 anchor 对象的序列
images 数组	封装页面中的图像信息
links 数组	HTML 文件中的超链接，通过它可以得到超链接的信息并可加以控制
方　法	**说　明**
write()	输出内容到 HTML 文件中
writeln()	输出内容到 HTML 文件中
open()	打开一个已存在的文件或创建一个新文件来写入内容
close()	关闭文件
clear()	清理文件中的内容

事件处理：无。

（9）Form 对象。功能：封装了网页中由<form>标记定义的表单信息。其属性和方法见表 B-9。

表 B-9　Form 对象

属　性	说　明
action	表单提交后启动的服务器应用程序的 URL，与 form 定义中的 action 属性相对应
name	表单的名称，与 form 定义中的 name 属性相对应
method	指出浏览器将信息发送到由 action 属性指定的服务器的方法，它只可能是 get 或 post。Form 对象的此属性对应 form 定义中的 method 属性

续表

属　性	说　明
target	指出服务器应用程序的执行结果返回的窗口，对应 form 定义中的 target 属性
encoding	指出被发送的数据的编码方式，对应 form 定义中的 enctype 属性
elements	是一个数组，其元素是表单的各个输入域对象
length	表单中输入域的个数
方　法	**说　明**
submit()	触发 Submit()事件，引起 onSubmit 事件处理的执行
reset()	清除表单中的所有输入，并将各输入域的值设为原来的默认值，该方法将触发 onReset 事件处理的执行

事件处理：onSubmit、onReset。

（10）History 对象。功能：历史清单对象，保存窗口或框架在某个时间段内访问的 URL 列表，并提供在列表中查找它们的方法。其属性和方法见表 B-10。

表 B-10　History 对象

属　性	说　明
current	当前历史项的 URL
length	历史列表中的项数
next	下一个历史项的 URL
previous	前一个历史项的 URL
方　法	**说　明**
back()	装载历史列表中的前一个 URL
forword()	装载历史列表中的下一个 URL
go()	该方法的参数可以是整数，也可以是字符串。当参数是整数 *i* 时，该方法将装载历史列表中与当前 URL 位置相距 *i* 的 URL，*i* 既可为正数，也可为负数。当参数是字符串时，该方法将装载历史列表中含该字符串的最近的 URL

事件处理：无。

（11）Location 对象。功能：用于存储当前的 URL 信息，可通过对该对象赋值来改变当前的 URL。其属性见表 B-11。

表 B-11　Location 对象

属　性	说　明
Hash	对应 Hash 数，即锚点名，如 #follow-up
Host	主机名或主机 IP 地址，如 www.njim.edu.cn
Hostname	是主机和端口的组合，如 www.njim.edu.cn:2000
Href	代表整个 URL
Pathname	是路径，在上例中是/java/index.html
Port	服务器端口号，如 2000
Protocol	代表协议，如 http
Search	查询信息，查询数据前加一个问号，这些数据包含在 URL 的最后一项

（12）Frame 对象。功能：一个 frame 对象对应一个<frame>标记定义。其属性和方法见表 B-12。

表 B-12　Frame 对象

属　性	说　明
name	框架的名称，对应<frame>定义中的 name 项
length	框架中包含的子框架数目
parent	包含当前框架的 Window 或 Frame
self	代表当前框架
top	指包含框架定义的最顶层窗口
window	与 self 含义相同
frames 数组	对应当前窗口中的所有框架
方　法	**说　明**
Focus()	使窗口获得焦点
Blur()	使窗口失去焦点
SetTimeout()	设置超时
ClearTimeout()	清除超时设置

事件处理：onBlur、onFocus、onLoad 和 onUnload。

2．函数

（1）eval()：参数 string 是一个字符串，该字符串的内容应是一个合法表达式。eval()函数将表达式求值，返回该值。

语法格式：eval(string)

（2）isNaN()：测试参数表达式的值是否为 NaN，若是，isNaN()返回 true；否则返回 false。

语法格式：isNaN(testValue)

（3）parseInt()：参数分析。参数 str 是一个字符串，可选参数 radix 是整数，若给出，则表示基数，若未给出，则表示基数为 10。parseInt()函数先对字符串形式的表达式求值，若求出的值是整数，则应转换为相应基数的数值。若不能求出整数值，则返回 NaN 或 0。

语法格式：parseInt(str[,radix])

（4）parseFloat()：参数分析。parseFloat()函数的使用与 parseInt()类似，其所求的值为浮点数。

语法格式：parseFloat(str)

附录 C　CSS 样式表属性

1．字体属性（见表 C-1）

表 C-1　字体属性

属　性	说　明
font-family	字体
font-size	字号
font-style	字体风格
font-weight	字加粗
font-variant	字体变化
font	字体综合设置

2．颜色和背景属性（见表 C-2）

表 C-2　颜色和背景属性

属　　性	说　　明
color	指定页面元素的前景色
background-color	指定页面元素的背景色
background-image	指定页面元素的背景图像
background-repeat	决定一个被指定的背景图像被重复的方式。默认值为 repeat
background-attachment	指定背景图像是否跟随页面内容滚动。默认值为 scroll
background-position	指定背景图像的位置
background	背景属性综合设定

3．文本属性（见表 C-3）

表 C-3　文本属性

属　　性	说　　明
letter-spacing	设定字符之间的间距
text-decoration	设定文本的修饰效果，line-through 是删除线，blink 是闪烁效果。默认值为 none
text-align	设置文本横向排列对齐方式
vertical-align	设定元素在纵向上的对齐方式
text-indent	设定块级元素第一行的缩进量
line-height	设定相邻两行的间距

4．方框属性（见表 C-4）

表 C-4　方框属性

属　　性	说　　明
Margin-top	设定 HTML 文件内容与块元素的上边界距离。值为百分比时参照其上级元素的设置值。默认值为 0
Margin-right	设定 HTML 文件内容与块元素的右边界距离
Margin-bottom	设定 HTML 文件内容与块元素的下边界距离
Margin-left	设定 HTML 文件内容与块元素的左边界距离
Margin	设定 HTML 文件内容与块元素的上、右、下、左边界距离。如果只给出 1 个值，则被应用于 4 个边界，如果只给出 2 个或 3 个值，则未显式给出值的边用其对边的设定值
padding-top	设定 HTML 文件内容与上边框之间的距离
padding-right	设定 HTML 文件内容与右边框之间的距离
padding-bottom	设定 HTML 文件内容与下边框之间的距离
padding-left	设定 HTML 文件内容与左边框之间的距离
padding	设定 HTML 文件内容与上、右、下、左边框的距离。设定值的个数与边框的对应关系同 margin 属性
border-top-width	设置元素上边框的宽度
border-right-width	设置元素右边框的宽度
border-bottom-width	设置元素下边框的宽度
border-left-width	设置元素左边框的宽度
border-width	设置元素上、右、下、左边框的宽度。设定值的个数与边框的对应关系同 margin 属性
border-top-color	设置元素上边框的颜色
border-right-color	设置元素右边框的颜色

续表

属　性	说　明
border-bottom-color	设置元素下边框的颜色
border-left-color	设置元素左边框的颜色
border-color	设置元素上、右、下、左边框的颜色。设定值的个数与边框的对应关系同 margin 属性
border-style	设定元素边框的样式。设定值的个数与边框的对应关系同 margin 属性。默认值为 none
border-top	设定元素上边框的宽度、样式和颜色
border-right	设定元素右边框的宽度、样式和颜色
border-bottom	设定元素下边框的宽度、样式和颜色
border-left	设定元素左边框的宽度、样式和颜色
width	设置元素的宽度
height	设置元素的高度
float	设置文字围绕于元素周围。left—元素靠左，文字围绕在元素右边；right—元素靠右，文字围绕在元素左边；none—以默认位置显示
clear	清除元素浮动。none—不取消浮动；left—文字左侧不能有浮动元素；right—文字右侧不能有浮动元素；both—文字两侧都不能有浮动元素

5．列表属性（见表 C-5）

表 C-5　列表属性

属　性	说　明
list-style-type	表项的项目符号。disc—实心圆点；circle—空心圆；square—实心方形；decimal—阿拉伯数字；lower-roman—小写罗马数字；upper-roman—大写罗马数字；lower-alpha—小写英文字母；upper-alpha—大写英文字母；none—不设定
list-style-image	用图像作为项目符号
list-style-position	设置项目符号是否在文字里面，与文字对齐
list-style	综合设置项目属性

6．定位属性（见表 C-6）

表 C-6　定位属性

属　性	说　明
top	设置元素与窗口上端的距离
left	设置元素与窗口左端的距离
position	设置元素位置的模式
z-index	z-index 将页面中的元素分成多个“层”，形成多个层“堆叠”的效果，从而营造出三维空间效果

附录 D　ASP.NET 对象的集合、属性、方法和事件

1．Page 对象（见表 D-1）

表 D-1　Page 对象

属　性	语　法	描　述
ClientTarget	Page. ClientTarget	客户端浏览器属性
ErrorPage	Page. ErrorPage= "filename"	当前网页发生未处理的异常时，将转向错误信息网页 Filename；若未设置此属性值，将显示默认错误信息网页

续表

属　性	语　法	描　述
IsPostBack	Page. IsPostBack	网页加载状况。值为 False，表示网页第一次加载；值为 True，表示响应客户端请求而被重新加载
IsValid	Page. IsValid	表示网页上的验证控件是否全部通过验证。值为 True，表示全部通过验证；值为 False，表示至少有一个验证失败
Visible	Page. Visible= BooleanValues（布尔值）	设置是否显示网页。True 显示，False 不显示
方　法	**语　法**	**描　述**
DataBind()	Page.DataBind()	将数据源与页面上的服务器控件进行绑定
Dispose()	Page.Dispose()	让服务器控件在释放内存前执行清理操作
FindControl ()	Page.FindControl(Id)	在页面上搜索标识为 Id 的服务器控件。若找到，返回该控件；若找不到，则返回 Nothing
HasControls()	Page. HasControls()	Page 对象中包含服务器控件返回 True，否则返回 False
MapPath ()	Page.MapPath(VirtualPath)	将虚拟路径 VirtualPath 转换为实际路径
事　件	**语　法**	**描　述**

DataBinding()	protected void Page_DataBinding (object sender, EventArgs e)
	{
	…
	}
		当网页上的服务器控件连接数据源时触发该事件

Disposed()	protected void Page_Disposed(object sender, EventArgs e)
	{
	…
	}
		网页从内存释放时触发该事件

Error()	protected void Page_Error(object sender, EventArgs e)
	{
	…
	}
		网页上发生未处理的异常情况时触发该事件

Init()	protected void Page_Init (object sender, EventArgs e)
	{
	…
	}
		当网页初始化时触发该事件。可以使用此事件来读取或初始化控件属性

Load()	protected void Page_Load(object sender, EventArgs e)
	{
	…
	}
		当加载网页时触发该事件

Unload()	protected void Page_UnLoad(object sender, EventArgs e)
	{
	…
	}
		网页完成处理工作被卸载时触发该事件

2．Application 对象（见表 D-2）

表 D-2　Application 对象

属　性	语　法	描　述
AllKeys	Application. AllKeys, Application. AllKeys(index)	AllKey 返回所有的变量名, AllKeys(index)返回下标为 index 的变量名
Count	Application.Count	获取 Application 对象变量的个数
Contents	Application. Contents	获取对 Application 对象的引用
方　法	**语　法**	**描　述**
Add()	Application.Add(name, value)	添加一个新的 Application 对象变量，名为 name，值为 value
Clear()	Application. Clear()	清除所有 Application 对象变量
Get()	Application. Get(name) Application. Get(index)	获取名称为 name 或下标为 index 的变量值
GetKey()	Application.GetKey(index)	获取索引值为 index 的变量名
Lock()	Application. Lock()	锁定。禁止其他用户修改 Application 对象变量
Remove()	Application. Remove(name)	清除名为 name 的变量
RemoveAll()	Application. RemoveAll()	清除所有的 Application 对象变量
RemoveAt()	Application. RemoveAt(index)	清除索引为 index 的变量
Set()	Application. Set(name,value)	将名为 name 的变量值修改为 value
UnLock()	Application. UnLock()	解除对 Application 对象变量的锁定
事　件	**语　法**	**描　述**

Application_Start()	Protected void application_Start(object sender, EventArgs e)
	{
	…
	}
		第一个用户访问该站点时触发

Application_End()	Protected void application_End(object sender, EventArgs e)
	{
	…
	}
		关闭 Web 服务器时触发

Application_BeginRequest()	Protected void Application_BeginRequest(object sender, EventArgs e)
	{
	…
	}
		在每一个 ASP.NET 被请求时触发

Application_EndRequest()	Protected void Application_EndRequest(object sender, EventArgs e)
	{
	…
	}
		结束 ASP.NET 程序时触发

3．Request 对象（见表 D-3）

表 D-3　Request 对象

集　合	语　法	描　述
QueryString	Request.QueryString[String 参数]	取回 URL 请求字符串
Form	Request.Form[String 参数]	取得客户端表格元素中所填入的信息
Cookies	Request.Cookies [String]	取得客户端浏览器的 Cookies 值
ServerVariable	Request.ServerVariables[服务器环境变量]	取得服务器端环境变量的值
Browser	Request.Browser.浏览器特性	获取客户端浏览器支持的功能信息
属　性	**语　法**	**描　述**
ApplicationPath	Request. ApplicationPath	获取 ASP.NET 应用的虚拟目录（URL）
HttpMethod	Request. HttpMethod	获取客户端使用的 HTTP 数据传输方式（Post、Get 或 Head）
Path	Request. Path	获取当前请求网页的虚拟路径和网页名称
TotalBytes	Request. TotalBytes	获取客户端请求数据的字节数
PhysicalPath	Request. PhysicalPath	获取当前请求网页的物理路径
Url	Request. Url	获取当前请求网页的 URL 完整信息
UserHostAddress	Request. UserHostAddress	获取客户端的 IP 地址
UserHostName	Request. UserHostName	获取客户端的 DNS 名称
方　法	**语　法**	**描　述**
Mappath()	Request.MapPath(VirtualPath)	将参数 VirtualPath 指定的虚拟路径映射为实际路径
SaveAs()	Request.SaveAs(filename, includeHeaders)	将 HTTP 请求保存到磁盘。其中参数 filename 是保存文件的路径及文件名，includeHeaders 指定是否保存 HTTP 标头

4．Response 对象（见表 D-4）

表 D-4　Response 对象

属　性	语　法	描　述
BufferOutput	Response. BufferOutput=BooleanValues（布尔值）	设置 HTTP 输出是否启用缓冲处理，默认为 True，启用
Charset	Response. Charset=字符编码	设置字符的编码方式
ContentType	Response. ContentType	获取或设置输出流的 HTTP MIME 类型
Expires	Response. Expires=minutes	获取或设置浏览器上缓存页过期之前的分钟数，如果用户在缓存之前返回同一页，则显示缓存的版本
IsClientConnected	Response. IsClientConnected	获取客户端是否与服务器保持连接的信息
方　法	**语　法**	**描　述**
Write()	Response.Write(String)	输出信息到客户端
Redirect()	Response.Redirect(String)	重定向客户端到另一个 URL 位置
ClearContent()	Response. ClearContent()	清除缓冲区的内容
End()	Response.End()	服务器立即停止处理脚本，并返回当时的状况
Flush()	Response.Flush()	立即把缓存在服务器端的 Response 输出信息送客户端显示
WriteFile()	Response.WriteFile(filename)	将指定文件的内容直接输出至客户端

5．Session 对象（见表 D-5）

表 D-5　Session 对象

属　性	语　法	描　述
SessionID	Session.SessionID	用于标识每个 Session 对象的标识码
Count	Session. Count	获取 Session 对象变量的个数
IsReadOnly	Session. IsReadOnly	该值指示会话是否为只读，默认值为 False
IsNewSession	Session. IsNewSession	该值指示会话是否与当前请求一起被创建
TimeOut	Session.Timeout[=Minutes]	设置 Session 对象的失效时间，单位为分钟，默认为 20 分钟
方　法	**语　法**	**描　述**
Abandon()	Session.Abandon()	用于删除所有存储在 Session 对象中的变量
Add()	Session.Add(name，value)	增加一个新的 Session 变量，名为 name，值为 value
Clear()	Session. Clear()	清除全部 Session 变量的值
Remove()	Session. Remove(name)	清除名称为 name 的 Session 变量
RemoveAll()	Session. RemoveAll()	清除所有的 Session 变量
RemoveAt()	Session. RemoveAt(index)	清除会话集合中下标为 index 的 Session 变量
事　件	**语　法**	**描　述**

Session_Start()	protected void Session_Start(object sender, EventArgs e)
	{
	…
	}
		在服务器创建新的会话时触发

Session_End()	protected void Session_End(object sender, EventArgs e)
	{
	…
	}
		在会话被放弃或超时时触发

6．Server 对象（见表 D-6）

表 D-6　Server 对象

属　性	语　法	描　述
MachineName	Server. MachineName	获取服务器端机器的名称，只读属性
ScriptTimeout	Server.ScriptTimeOut=n	获取和设置程序执行的最长时间，即程序必须在该时间段内执行完毕。单位为秒，系统默认值为 90 秒
方　法	**语　法**	**描　述**
CreateObject()	Server.CreateObject("ProgID")	创建 COM 对象的一个服务器实例
Execute()	Server. Execute(path)	执行由 path 指定的 ASP.NET 程序, 执行完毕后仍继续原程序的执行
HtmlEncode()	Server. HtmlEncode(string)	对要在浏览器中显示的字符串 string 进行编码
HtmlDecode()	Server. HtmlDecode(string)	与 HtmlEncode 相反，还原为原来的字符串
MapPath()	Server.MapPath(String)	将参数 path 指定的虚拟路径转换为物理路径
Transfer()	Server. Transfer(path)	结束当前 ASP.NET 程序的执行，并开始执行参数 path 指定的程序
UrlEncode()	Server. UrlEncode(string)	对字符串 string 以 Url 格式进行编码
UrlDecode()	Server. UrlDecode(string)	对 Url 格式字符串进行解码

参 考 文 献

1　吉根林，顾韵华．Web 程序设计（第 3 版）．北京：电子工业出版社，2011．

2　张昌龙，辛永平．ASP.NET 4.0 从入门到精通．北京：机械工业出版社，2011．

3　郑阿奇．ASP.NET 4.0 实用教程（第 2 版）．北京：电子工业出版社，2013．

4　张正礼．ASP.NET 4.0 网站开发与项目实战．北京：清华大学出版社，2012．

5　Robert W·Sebesta．Web 程序设计．北京：清华大学出版社，2010．

6　杨树林，胡洁萍．ASP.NET 程序设计案例教程．北京：人民邮电出版社，2011．

7　梁爽等．.NET 框架程序设计．北京：清华大学出版社，2010．

8　韩颖等．ASP.NET 3.5 动态网站开发基础教程．北京：清华大学出版社，2010．

9　（美）麦克唐纳，弗里曼，兹普兹塔．ASP.NET 4 高级程序设计（第 4 版）．博思工作室，译．北京：人民邮电出版社，2011．

10　陈作聪．Web 程序设计——ASP.NET 上机实验指导．北京：清华大学出版社，2012．

反侵权盗版声明

举报电话：（010）88254396；（010）88258888

传　　真：（010）88254397

E-mail：　dbqq@phei.com.cn

通信地址：北京市万寿路 173 信箱

　　电子工业出版社总编办公室

邮　　编：100036